Schweizerische
Fachschule
TEKO

DIPLOMARBEIT

Elektrotechnik HF 2022

Smarte Parkplatzüberwachung

Lehrinstitut:	**TEKO Schweizerische Fachschule Zürich**
Studiengang:	**Elektrotechnik HF**
Diplomand:	**Dominik Viola**
Klasse:	**Z-TEL-19-T-a**
Klassifizierung:	**Öffentlich**
Ort, Datum:	**Glattbrugg, Oktober 2022**

Diplomarbeit HF Elektrotechnik 2022
Smarte Parkplatzüberwachung

Diplomand
Dominik Viola
Glatthofstrasse 12
8152 Glattbrugg
T: 076 456 06 96
E: dominik-viola@bluewin.ch

Diplomcoach
Christian Meier
TEKO Schweizerische Fachschule AG
E: christian.meier@edu.teko.ch

Auftraggeberschaft und Diplomexperte
TEKO Zürich
Adrian Aegler
Europa-Strasse 18
8152 Glattbrugg
T: 043 305 23 37
E: adrian.aegler@teko.ch

Glattbrugg, Oktober 2022

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe.

Die wörtlich oder inhaltlich den im Literaturverzeichnis aufgeführten Quellen und Hilfsmitteln entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht.

Diese Diplomarbeit ist noch nicht veröffentlicht worden. Sie ist somit weder anderen Interessenten zugänglich gemacht noch einer anderen Prüfungsbehörde vorgelegt worden.

Ort, Datum

8152 Glattbrugg ZH, Oktober 2022

Unterschrift

Vorwort

Die Idee für diese Diplomarbeit ist von Adrian Aegler als mögliche Option genannt worden, nachdem meine erste eingegebene Diplomarbeit durch die Firma, für die ich tätig bin, abgesagt wurde.
Die Idee von Adrian Aegler hat mir sofort gefallen und ich wusste das ich das Erlernte über die sechs Semester an der TEKO in dieser Arbeit umsetzen kann.
Diese Arbeit hat viele Aspekte, welche in der heutigen Zeit wichtig geworden sind.
Zum einen die smarte Kommunikation mit Geräten genannt ,,Internet der Dinge'', wie auch die Energiegewinnung mit Solar.

Diese Diplomarbeit wird durch mich allein durchgeführt und dokumentiert. Die Ausarbeitung der möglichen Lösungen wird durch Versuche ausgearbeitet und nach der Entscheidung realisiert sowie in Betrieb genommen.

Ich möchte mich herzlich bei Adrian Aegler und der TEKO für die Chance bedanken, dass ich für sie dieses Projekt realisieren durfte.
Ich danke meinem Diplomcoach Christian Meier für seine Unterstützung bei dieser Diplomarbeit.
Einen grossen Dank möchte ich an Jenny Meier richten, für die Überprüfung der Rechtschreibung.
Ein weiterer Dank geht an Florian Meier, der mir mit seinem Fachwissen im Programmieren den ein oder anderen Tipp geben konnte und ich dadurch mein Wissen erweitern konnte und das Projekt erfolgreich abschliessen konnte.
Der letzte Dank geht an meinen Vater Felix Viola, der mir geholfen hat, denn Aufbau bei der TEKO aufzustellen.

Management Summary

In diesem Projekt geht es darum für die TEKO Zürich eine geeignete Visualisierung der Parkplatzbelegung der Dozentenparkplätze an der Europa-Strasse 18 in Opfikon zu entwickeln.

Dieses Projekt soll für das Sekretariat eine Erleichterung sein, sodass diese nicht zu den Parkplätzen gehen müssen, um zu sehen, ob diese belegt sind.

Ziel ist es, über geeignete Sensoren die Parkplätze zu überprüfen. Diese sollen die Werte ins Sekretariat übertragen und dort auf einem Tablett oder Bildschirm anzeigen, welcher der sechs Parkplätze belegt ist. Ausserdem soll das System sich selbst versorgen können, was mit einer Solaranlage erreicht werden sollte.

Mit Messversuchen soll herausgefunden werden, welches die beste Lösung für diesen Standort ist.

Drei Optionen von Sensoren stehen zur Verfügung, welche in einem günstigeren Preisrahmen liegen und wasserdicht sind. Option 1 ist ein Ultraschallsensor, welcher über UART kommuniziert. Die zweite Option ist ein Lasersensor, welcher über UART oder RS485 kommuniziert. Option drei ist ein weiterer Ultraschallsensor, welcher über digitale Ein- und Ausgänge kommuniziert.
Alle Optionen sind mit einem Mikrokontroller verbunden. Dieser sendet über das TEKO WLAN die Daten an einen Raspberry Pi. Auf dem Raspberry Pi ist mit Node-RED eine Benutzeroberfläche erstellt, wo über die IP-Adresse des Raspberry darauf zugegriffen werden kann. Dazu kommt, dass alle Optionen eine Solaranlage haben.

Die Optionen wurden in mehreren Testversuchen evaluiert und auf das Beste Preis- / Leistungsverhältnis geprüft.

Die Empfehlung liegt bei der dritten Option. Diese beinhaltet ein Ultraschallsensor, der aus einem Sensor und einem Sensorboard besteht. Das Sensorboard kann über digitale Ein- und Ausgänge gesteuert werden. Das Preis- / Leistungsverhältnis ist am besten. Der Stromverbrauch ist mit dieser Option am geringsten, was auch bei schlechterem Wetter mit der Solaranlage zu vereinbaren ist, ohne zusätzliche externe Speisungen.
Das komplette Budget für das Material beträgt 1892.90 Schweizer Franken.

Inhaltsverzeichnis

Ehrenwörtliche Erklärung 3
Vorwort 4
Management Summary 5
Vorgehen 9
Informationen 9
Planung 9
Entscheidung 10
Realisierung 10
Kontrolle 10
Auswertung 10
Informationen 11
Planung 12
Pflichtenheft 12
Funktionale Anforderungen 12
Nicht funktionale Anforderungen 13
Projektstrukturplan 14
Zeitplan 14
Meilensteine 14
Risikoanalyse 15
Entscheidung 16
Sensoren für die Distanzmessung 16
Auswahl der Sensoren 16
Testversuch mit Ultraschallsensor A02YYUW 17
Equipment 17
Testversuch 18
Ergebnis 18
Testversuch mit dem Laser Distanz Sensor SEN0491 20
Equipment 20
Testversuch 21
Ergebnis 21
Testversuch mit Ultraschallsensor JSN-SR04T-V3.0 29
Equipment 29
Testversuch 30
Ergebnis 30
Entscheidung zu den Sensoren 32

Mikrokontroller....33
Auswahl des Mikrokontrollers....33
Testversuch mit dem Arduino Mega 2560....34
Equipment und Messaufbau....35
Ergebnis....35
Testversuch mit dem ESP8266....36
Equipment und Messaufbau....38
Ergebnis....38
Testversuch mit dem ESP32 S2 Saola 1R Developmentboard....39
Equipment und Messaufbau....40
Ergebnis....40
Entscheidung zu den Mikrokontrollern....41
Raspberry Pi....42
Equipment und Messaufbau....43
Ergebnis....43
I/O Erweiterung....44
Equipment und Messaufbau....45
Ergebnis und Entscheid....45
Temperatursensor....46
Equipment und Messaufbau....46
Ergebnis und Entscheid....47
Ventilator....48
Equipment und Messaufbau....49
Ergebnis und Entscheid....49
Solar....50
Equipment und Messaufbau....52
Ergebnis....53
Entscheid....54
Realisierung....55
Software des ESP32....55
Testaufbau für die Software des ESP32....56
Flussdiagramm der Software des ESP32....57
Softwarecode des ESP32 und die einzelnen Funktionen....58
Softwarecode Sensor Parkplatz....59
Softwarecode Temperatur....60
Softwarecode Zeit....61

Node-RED 62
Node-RED und die einzelnen Funktionen 63
Node-RED Parkplatz 1 bis 6 64
Node-RED ESP32 aktiv 66
Node-RED Temperatur und Ventilatoren 68
Node-RED Zeit 71
Der Prototyp 72
Elektroschema des Prototyps 73
Sensorbox 74
Sensorboardbox 76
Elektronikbox 80
Solarpanel 86
Ergebnis 87
Der Aufbau bei der TEKO Zürich 88
Die Halter bei der TEKO Zürich 88
Die Verdrahtung bei der TEKO Zürich 90
Nummerierung der Parkplätze bei der TEKO Zürich 93
Kontrolle 94
Inbetriebnahmeprotokoll 94
Ergebnis der Inbetriebnahme 95
Auswertung 96
Persönliches Resümee 96
Abbildungsverzeichnis 97
Tabellenverzeichnis 99
Linkverzeichnis 100
Anhängeverzeichnis 100

Vorgehen

Die IPERKA-Projektmethode geht davon aus, dass jedes Projekt in sechs Phasen unterteilt wird, in denen einige Aktivitäten durchgeführt und Fragen geklärt werden müssen. Diese Anwendung, verlangt eine strukturierte Vorgehensweise.
Einen Teil davon wurde von der Internetseite *https://jovanovicmilos.wordpress.com/iperka* übernommen und auf dieses Projekt angepasst.

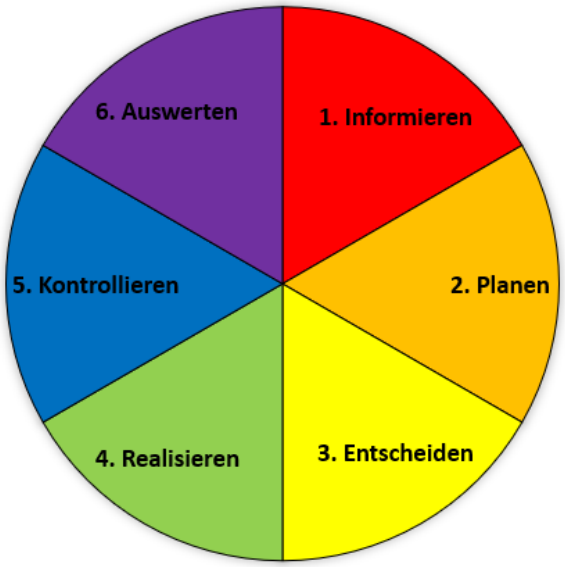


Abbildung 1: Projektablauf nach IPERKA

Informationen

Der erste Schritt besteht darin, Informationen systematisch zu sammeln.

- Wie sieht der Auftrag aus?
- Welche Vorgaben gibt es?
- Bis wann ist dieser Auftrag fertig zu stellen?
- Welche Informationen sind, wichtig und wo gibt es diese?
- Wer ist für was zuständig?
- Welche Ziele werden angestrebt?

Planung

Für das Projekt werden mögliche Lösungsvarianten und Vorgehensweisen erarbeitet.

- Wie sieht die Vorgehensweise aus?
- Welche Hilfsmittel, Geräte werden gebraucht?
- Welche Probleme könnten auftreten?
- Wie ist mit überraschenden Problemen/ Risiken umzugehen?

Entscheidung

Nach der Planung wurde eine Lösungsvariante gewählt. Entscheidungen sollten nicht hinausgezögert werden, da dies in der Regel lange dauert. Nachdem die Entscheidung getroffen wurde, wurden die folgenden Faktoren identifiziert:

- Welche Vorgehensweise wird gewählt?
- Verantwortlichkeiten festlegen
- Sind die Ideen sinnvoll?
- Wie kann man das Risiko reduzieren?

Realisierung

Zeitlich nimmt die Umsetzung bzw. Durchführung meist einen grossen Teil eines Projektes ein. Deshalb werden einzelne Arbeitsschritte eingeplant. Der erstellte Zeitplan ist einzuhalten und sollte nicht ohne zwingenden Grund geändert werden.

- Wie kann die Einhaltung des Zeitplanes überwacht werden?
- Wie wird überprüft, ob alles vorhanden und bereit ist?
- Wann wird mit der Umsetzung begonnen?

Kontrolle

Jeder durchgeführte Schritt muss überprüft werden. Kontrollieren bedeutet zum Beispiel, den Schritt noch einmal auszuführen, durchzulesen und darüber nachzudenken.

- Wurde das Ziel erreicht?
- Wurde der Zeitplan eingehalten?
- Wurden alle Aufgaben erfüllt?
- Wurde eine Inbetriebnahme durchgeführt mit dem Kunden?

Auswertung

Den gesamten Prozess des Projekts noch einmal in Gedanken durch und bewerten und auswerten.

- Welche Probleme wurden gelöst?
- Was war positiv, was negativ?
- Hat der Zeitplan funktioniert?
- Gibt es Dinge, die verbessert werden können/müssen?
- Welche Schwierigkeiten gab es?

Informationen

Wie sieht der Auftrag aus?

Der Auftraggeber des Projektes ist die TEKO Schule in Zürich. Für die Dozentenparkplätze soll eine Lösung erarbeitet werden, dass im Sekretariat erkennt, werden kann, ob die Parkplätze besetzt oder frei sind. Diese Lösung soll sich mit Solar selbst versorgen können.

Welche Vorgaben gibt es?

Es gibt auf dem Parkplatz sechs Parkplätze, welche für die Dozenten der TEKO sind. Wenn möglich soll das System von 21:00 bis 07:00 Uhr in einen Stromsparmodus schalten. Es ist wünschenswert, wenn die Spannungsversorgung über Solar laufen würde. An der Wand auf dem Parkplatz darf kein Solarpanel montiert werden, da diese der SBB gehört. In der Eingabe der Diplomarbeit wurde erwähnt, dass mit einem Raspberry Pi und einem Mikrokontroller die Lösung erarbeitet wird.

Bis wann ist dieser Auftrag fertig zu stellen?

Der Auftrag ist bis spätestens zum 26. Oktober 2022 fertig zu stellen.

Welche Informationen sind, wichtig und wo gibt es diese?

Alle wichtigen Informationen sind beim Schulleiter Adrian Aegler zu beziehen.

Wer ist für was zuständig?

Für das ganze Projekt ist Dominik Viola zuständig.
Für die Einbindung des Raspberry Pi ins TEKO Zürich WLAN ist Joe Brändli verantwortlich.

Welche Ziele werden angestrebt?

Angestrebt wird das ganze Projekt mit den Soll-Zielen zu erreichen und der TEKO Zürich eine zuverlässige Lösung zu erarbeiten.

Planung

Pflichtenheft

In diesem Projekt soll ein vollautomatisiertes Parkplatzüberwachungssystem erarbeitet werden, welches folgende Grobanforderungen erfüllt.

Funktionale Anforderungen

ID	Titel	Beschreibung	Kategorie
01	Sprache	Die Ausgaben der Software über die serielle Schnittstelle ist in der Sprache Deutsch.	muss
02	Node-RED	Das Parkplatzsystem soll auf einem Tablett anzeigen welche Parkplätze belegt sind. Die Parkplätze müssen einzeln beschriftet sein und klar durch Farbe erkennbar sein, welcher Platz belegt und welcher frei ist. Ausserdem soll auf der Node-RED Oberfläche die Gehäusetemperatur angezeigt werden.	muss
03	Sensoren (Parkplatzerkennung)	Die Sensoren erkennen, ob ein Fahrzeug auf einem der Parkplätze steht, geben einem Mikrokontroller einen Wert zurück, sodass dieser via Raspberry Pi die Daten an die Node-RED Oberfläche sendet.	muss
04	Sensor (Temperatur)	Der Temperatursensor erkennt die aktuelle Temperatur im Gehäuse, dieser soll vor einer Überhitzung des Systems schützen.	muss
05	Lüfter	Die Lüfter sind im Gehäuse der Steuerung eingebaut und müssen sich bei zu hoher Temperatur einschalten.	muss
06	Solar	Das Projekt soll mit Solar betrieben werden.	soll
07	Strom sparen	Ausschaltung in der Nacht von 21:00 – 07:00 Uhr	soll
08	Ablage, Verwaltung und Versionierung	Das Entwicklungsprojekt für das Parkplatzsystem soll auf einem netzbasierten Dienst zur Verwaltung für Software abgelegt werden. Z.B GitHub.	soll

Tabelle 1: Funktionale Anforderungen

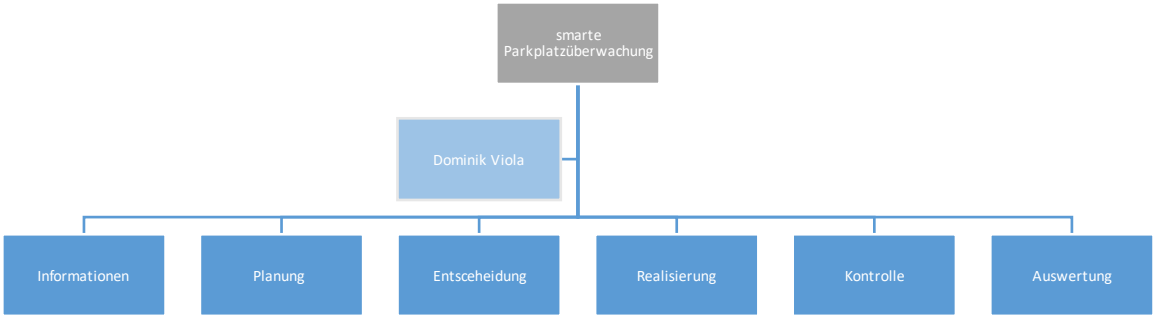
Nicht funktionale Anforderungen

Kategorie	Beschreibung	Einstufung
Skalierbarkeit	Das Parkplatzsystem soll durch das Anschliessen von weiteren Sensoren erweiterbar und modifizierbar sein.	hoch
	Alle Fehlermeldungen sollen informativ und klar sein.	mittel
Zuverlässigkeit	Das Parkplatzsystem sollte zuverlässig und täglich nutzbar sein und dies bei jeder Wetterlage.	hoch
Notfunktion	Sollte die automatische Lüftersteuerung nicht funktionieren, kann der Benutzer selbst via Node-RED die Lüfter ansteuern.	hoch

Tabelle 2: Nicht funktionale Anforderungen

Projektstrukturplan

Der Projektstrukturplan zeigt die Hierarchie in diesem Projekt.



Zeitplan

Der Zeitplan wird mit der Software Gantt erstellt und ist unter Anhänge einzusehen.

Meilensteine

Meilensteine sind die wichtigsten Ereignisse dieses Projektes. Mit ihnen startet und endet ein Abschnitt des Projektes.

Freigabe der Diplomarbeit: 15.07.2022 Beginn der Diplomarbeit
Informationen sammeln: 15.07.2022 Informationen beschaffen
Planung: 19.07.2022 Grober Zeitplan erstellen
Entscheidung: 22.07.2022 Beginn mit Versuchen zur Entscheidungsfindung
Realisierung: 24.08.2022 Beginn mit der Realisierung (Prototyp) und (Aufbau TEKO)
Kontrolle: 18.10.2022 Kontrolle des Aufbaus in der TEKO
Auswertung: 19.10.2022
Abgabe der Diplomarbeit: 26.10.2022
Präsentation der Diplomarbeit: 14.11.2022

Risikoanalyse

Die Risikoanalyse ist wichtig für das Projekt um die Gefahren wie auch die Chancen einzuschätzen. Dadurch kann auf eine Gefahr früh genug reagiert werden.

	Eintrittswahrscheinlichkeit des negativen Ereignisses		
Ausmass	gering	mittel	hoch
hoch		3	1
mittel			2
gering		4	

Tabelle 3: Risikoanalyse

Punkt 1 ist die aktuelle Bauteilknappheit. Dieser Punkt wird mit hoch im Ausmass sowie in der Eintrittswahrscheinlichkeit bewertet. Hier muss unbedingt bei den kleinsten Anzeichen ein Plan B gesucht werden, um die Auswirkung klein zu halten.

Punkt 2 ist die Firma in der, der Projektleiter angestellt ist. Hier könnte es sein ,dass dieser für die Firma eine Mehrarbeit leisten muss. Die Eintrittswahrscheinlichkeit ist hoch aber das Ausmass mittel. Dies kann zur Folge haben, dass einzelne Teilschritte des Projektes verzögert gestartet werden können.

Punkt 3 ist die Möglichkeit, dass beim Programmieren des Mikrokontrollers ein kleiner Fehler grosse Auswirkungen haben kann. Die Eintrittswahrscheinlichkeit ist mit mittel anzunehmen, aber das Ausmass hoch.

Punkt 4 ist, dass ein Bauteil defekt ankommt oder bei der Inbetriebnahme defekt wird. Die Eintrittswahrscheinlichkeit wird mit mittel angenommen und das Ausmass gering. Das Ausmass kann gering gehalten werden, indem bei nicht kostspieligen Bauteilen ein Stück mehr bestellt wird.

Entscheidung

Sensoren für die Distanzmessung

Auswahl der Sensoren

Für dieses Projekt wurde in den Parkhäusern des Flughafen Zürich sowie in der IKEA in Dietlikon geschaut, mit welchen Sensoren diese beiden Parkhäuser arbeiten.
Beide Parkhäuser arbeiten mit Ultraschallsensoren, was ein guter Hinweis war in welche Richtung es gehen kann.
Der Unterschied zu den beiden Parkhäusern ist, dass dieses Projekt im Freien installiert wird.

Die Sensoren müssen daher als zusätzliche Eigenschaft wasserdicht sein.

Deshalb wurde nach Sensoren gesucht, welche diese Anforderungen erfüllen.
Die Sensoren sollen auch einen geringen Stromverbrauch aufweisen, da das Projekt mit Solar betrieben werden soll. Der Preis sollte in einem überschaubaren Rahmen gehalten werden.
Mit diesen Eigenschaften wird die Auswahl dezimiert. Aus diesem Grund wurden bei der Firma Bastelgarage.ch drei verschiedene wasserdichte Distanzsensoren bestellt.
Diese werden in mehreren Testszenarien geprüft, sodass gesagt werden kann, welcher dieser Sensortypen der Beste für das Projekt ist.

Die Sensoren werden auf Ihre Kompatibilität, ihren Stromverbrauch sowie ihre weiteren Eigenschaften geprüft.

Bestellt wurden beim Onlineshop bastelgarage.ch ein Ultraschallsensor des Typs A02YYUW, ein Ultraschallsensor des Typs JSN-SR04T-V3.0 sowie ein Laser Distanz Sensor des Typs SEN0491.

Testversuch mit Ultraschallsensor A02YYUW

Der erste Sensor ist ein Ultraschallsensor, welcher über UART kommuniziert.

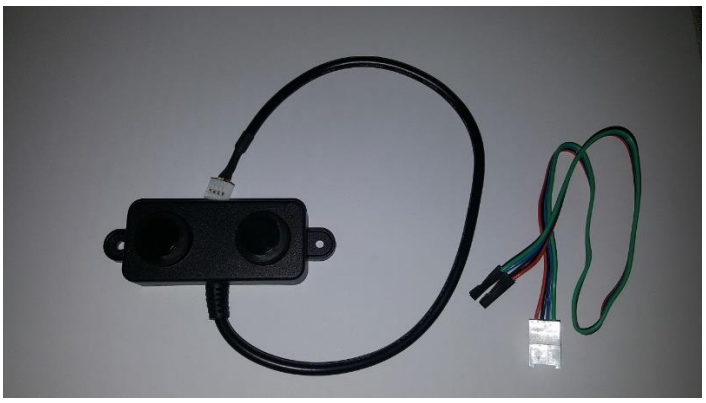


Abbildung 2: Ultraschallsensor A02YYUW

Dieser Sensor kann bei 3.3V und 5V betrieben werden. Der Sensor könnte direkt mit zwei Schrauben an eine Wand geschraubt werden. Dies würde weitere zusätzliche Kosten für ein Gehäuse einsparen.

Mit einem Versuchsaufbau wird der Sensor getestet.
Folgendes Equipment wird dafür verwendet.

Equipment

1x A02YYUW Ultraschallsensor wasserfest von DFRobot
1x ESP8266 NodeMCU V3 von ESPRESSIF
1x ESP8266 NodeMCU V3 Base Board Protoshield mit Spannungsregler von bastelgarage.ch
1x Laptop mit Arduino IDE
1x Fluke 179 Multimeter als Amperemeter

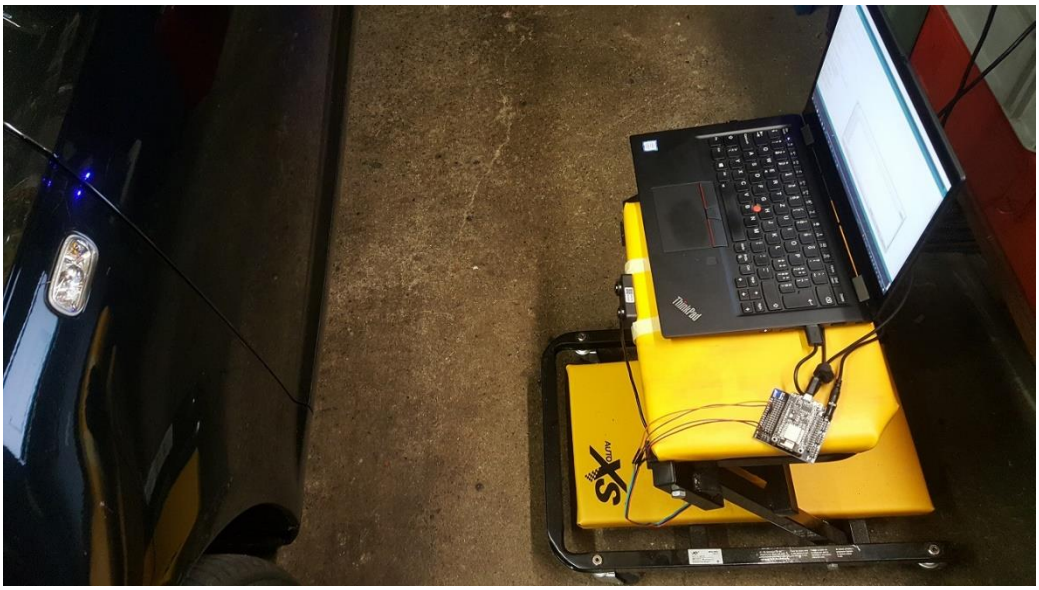


Abbildung 3: Messaufbau mit ESP8266 und Sensor

Testversuch

Zuerst wird der Softwarecode für den ESP8266, welcher auf der Homepage von dfrobot.com verfügbar ist, bearbeitet, für die Testzwecke erweitert und auf den ESP8266 geladen.
Der Sensor wird mit dem ESP8266 verbunden und getestet. Die Stromaufnahme wird bei 3.3V und bei 5V durchgeführt. Wichtig ist hier noch darauf zu achten, dass das RX und TX Signal vom Sensor gekreuzt am Mikrokontroller angehängt werden müssen.

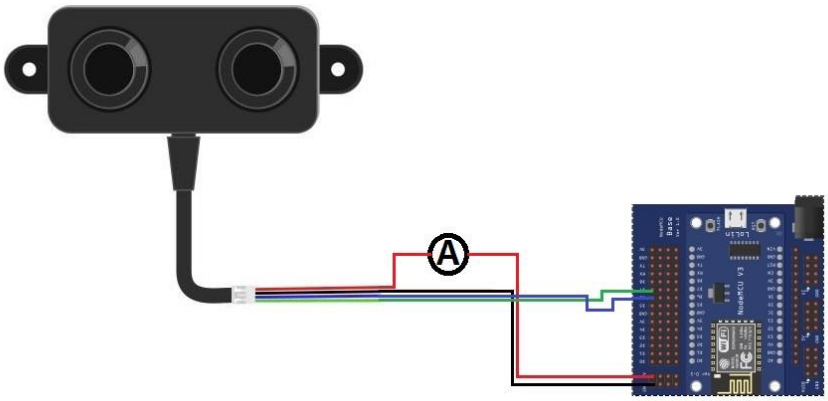


Abbildung 4: Schema ESP8266 und Sensor bei 3.3V

Ergebnis

Der Ultraschall Sensor erfüllt das gewünschte Ziel und detektiert den Abstand auf zwei bis drei Millimeter genau. Die spezifisch entwickelte Testsoftware soll ab einem Abstand kleiner gleich 50cm den Parkplatz als besetzt melden.

Abbildung 5: Parkplatz frei

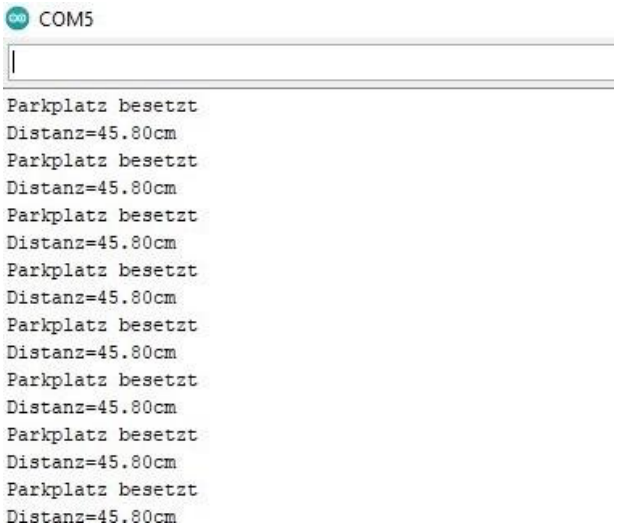


Abbildung 6: Parkplatz besetzt

Die Stromwerte des Sensors sind bei 3.3V für Ruhe (kein Ansprechen durch die Software) 5.25mA und bei Betrieb (Ansprechen durch die Software) 5.42mA.
Die Stromwerte bei 5V sind für Ruhe 6.94mA und bei Betrieb 7.08mA.

Da dies ein Ultraschallsensor ist, hat er keine Probleme mit den Farben der Fahrzeuge.

Dieser Sensor kommuniziert mit der Baudrate von 9600 Bit/s, es gibt Probleme beim Terminalprogramm, wenn andere Sensoren oder auch der ESP8266 selbst auf der Baudrate 9600 Bit/s kommunizieren.
Da die Applikation sechs dieser Sensoren bräuchte fällt dieser Sensor weg, auch wenn dieser alles andere erfüllen würde.

Testversuch mit dem Laser Distanz Sensor SEN0491

Der zweite Sensor ist ein Laser Distanz Sensor, welcher über UART kommuniziert.

Abbildung 7: Laser Distanz Sensor

Dieser Sensor kann zwischen 5-36V betrieben werden.
Der Sensor müsste in ein Gehäuse eingebaut werden, was zusätzliche Kosten verursachen würde.

Mit einem Versuchsaufbau wird der Sensor getestet.
Folgendes Equipment wird dafür verwendet.

Equipment

1x Laser Distanz Sensor wasserfest von DFRobot
1x Arduino Mega 2560 von Arduino
1x Laptop mit Arduino IDE
1x Fluke 179 Multimeter als Amperemeter

Abbildung 8: Messaufbau mit Arduino und Sensor

Testversuch

Zuerst wird der Softwarecode für den Arduino Mega 2560, welcher auf der Homepage von dfrobot.com verfügbar ist, bearbeitet, für die Testzwecke erweitert und auf den Arduino Mega 2560 geladen.
Der Sensor wird mit dem Arduino Mega 2560 verbunden und getestet. Die Stromaufnahme wird bei 5V durchgeführt. Auch bei diesem Testversuch ist es wichtig darauf zu achten, dass das RX und TX Signal vom Sensor gekreuzt am Mikrokontroller angehängt werden muss.

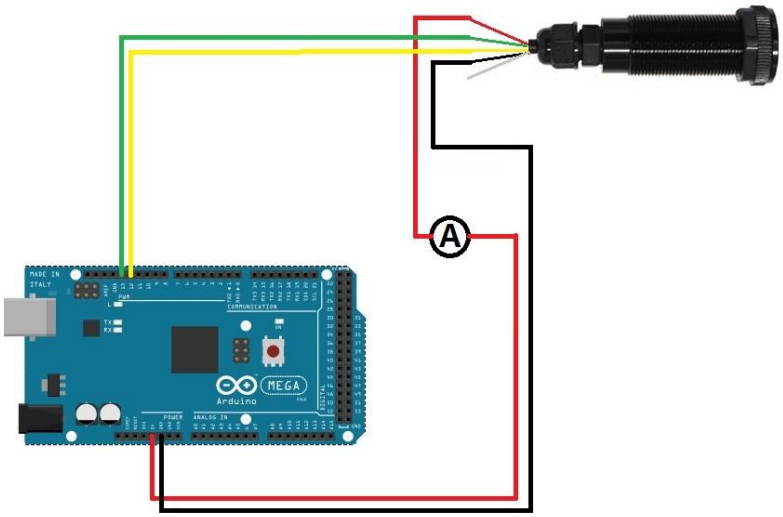


Abbildung 9: Schema Arduino Mega 2560 und Sensor

Ergebnis

Bei der ersten Inbetriebnahme wurden keine Daten vom Sensor gesendet.
Weil der Sensor keine Daten lieferte, musste mit der Fehlersuche begonnen werden.
Nach vielen Stunden im Internet surfen und informieren wurden die beiden Wandler «6 in 1 USB zu Serial» und «USB zu RS485», welche auf der DFRobot Seite zu diesem Sensor angegeben wurden, bestellt.

Abbildung 10: 6 in 1 USB zu Serial-Wandler

Abbildung 11: USB zu RS485-Wandler

Als die beiden Wandler da waren wurde der USB zu Serial Wandler ausprobiert.
Dies hatte immer noch keinen Erfolg gebracht, da gemäss der DFRobot Seite dieser Sensor automatisch mit der Software Serial Debug Assistent die Daten senden sollte.
Durch weiteres Suchen im Internet wurde entdeckt, dass es diesen Sensor in zwei verschiedenen Varianten gibt. Einmal geführt bei DFRobot mit der Artikelnummer SEN0491 als UART Variante und einmal unter SEN0492 als RS485 Variante.
Da auch die Dokumentation zu den beiden Sensoren sehr dürftig ist, sowie auch keine Projekte im Internet dazu zu finden sind, kam der Entschluss, die RS485 Variante, anstelle der UART Variante, auszuprobieren.

Der vorhandene Laser Distanz Sensor wurde mit dem USB zu RS485 Stick verbunden.

Da dieser Sensor dann mit dem RS485 Protokoll arbeitet, musste zuerst eine geeignete Software gefunden werden, welche erlaubt den gewünschten Befehl an den Sensor zu senden.
Nach ein wenig Recherche konnte ein erster Versuch gestartet werden mit der Software Device Monitoring Studio, welche eine Freeware ist und für 14 Tage alle Optionen freigeschaltet hat.

Jetzt konnte das erste Mal der Lese Befehl gesendet werden, welcher in der Dokumentation des RS485 Sensors steht.

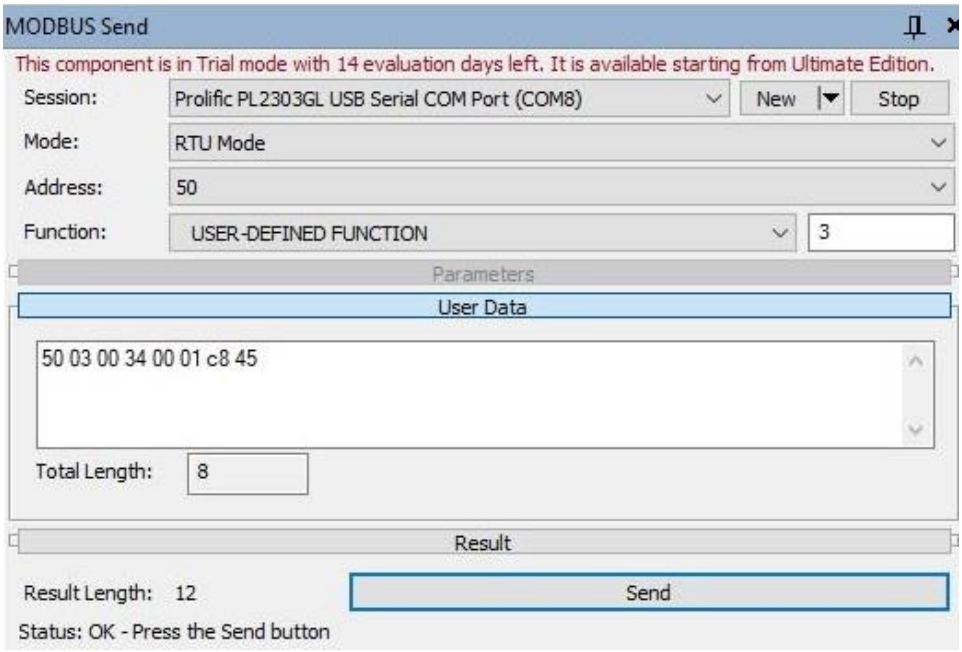


Abbildung 12: Software mit Befehl

Der Befehl, um die gemessenen Daten zu erhalten ist 50 03 00 34 00 01 C8 45 und setzt sich aus unterem Ausschnitt zusammen.

Slave address	Function code	Register address high bit	Register address low bit	Read length high bit	Read length low bit	CRC check high bit	CRC check low bit
0x50	0x03	RegH	RegL	LenH	LenL	CRCH	CRCL

Abbildung 13: Aufbau des Lesebefehles

Der Rückgabewert ist in HEX und muss nach unterem Ausschnitt entpackt werden.

Slave address	Function code	Data length	Data bit 1	Data bit 2	CRC check high bit	CRC check low bit
0x50	0x03	LenH	DataH	DataL	CRCH	CRCL

Abbildung 14: Aufbau des Rückgabewertes

Abbildung 15: Ausgabe der Software

Bekommt man jetzt folgende Werte zurück von der Software, kann man diese nach dem Schema auf der Seite von DFRobot zu diesem Sensor entschlüsseln.
0x50 = Modbus Adresse des Sensors
0x03 = Ist das Lese Kommando würde hier 0x06 stehen wäre es das Schreib Kommando
0x02 = Datenlänge
0x08FB = Ist die gemessene Länge in HEX und entspricht in Dezimal umgerechnet 2299mm.
Dies kann mit allen Antworten des Sensors gemacht werden.
Die hinteren beiden Hex Zahlen sind das high und low check bit.

Nach ein paar Versuchen erlaubte die Software keine weiteren Versuche und der USB-Wandler musste ausgesteckt werden und alles neu gestartet werden. Dieser Fehler kam sowohl mit dem 6 in 1 USB-Wandler sowie mit dem USB zu RS485 Wandler vor. Somit lag dieser Fehler definitiv an der Freeversion der Software.

Zu diesem Zeitpunkt war klar, dass die falsche Version von der bastelgarage.ch geliefert wurde. Der Sensor liefert Daten über die RS485 Software jedoch nicht über die UART Schnittstelle am Arduino. Aus Interesse wurde der Sensor auseinandergebaut.

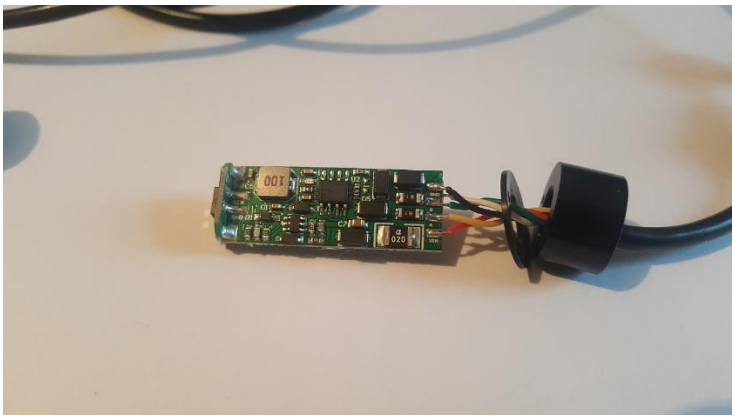


Abbildung 16: Laser Distanz Sensor RS485 Seite Top

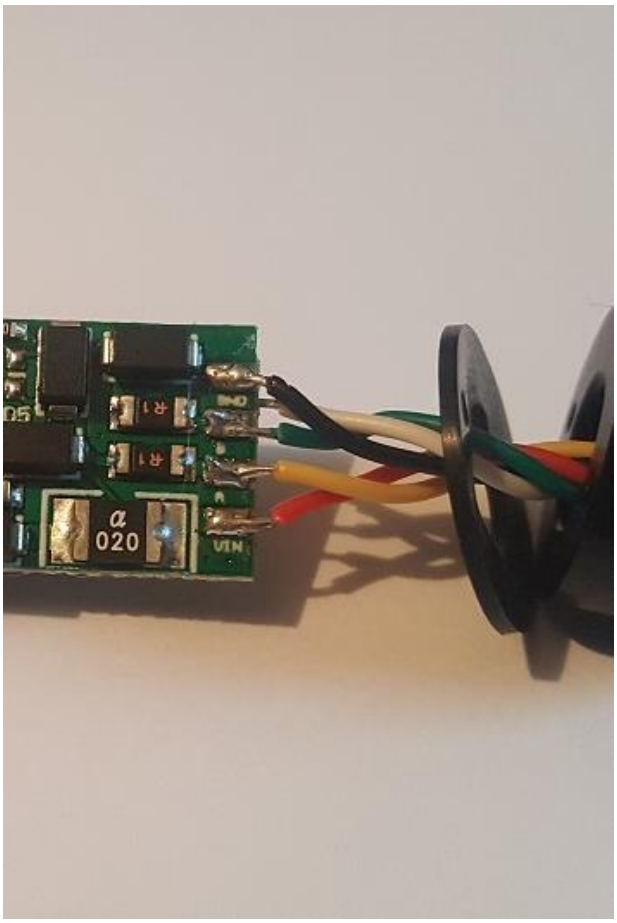


Abbildung 17: Grossaufnahme des Sensors RS485

In Abbildung 17 ist klar zu erkennen, dass dieser Sensor nach RS485 verkabelt wurde.

Die Signale A und B sind verbunden, was bei UART RX und TX sein müssten.

Auf der Bottom Seite des Prints gibt es Pins für RX und TX was die Signale für UART sind. Es kam die Idee nochmals einen Sensor zu bestellen und sollte dieser wieder auf RS485 verkabelt sein, wird dieser auf RX und TX um gelötet.

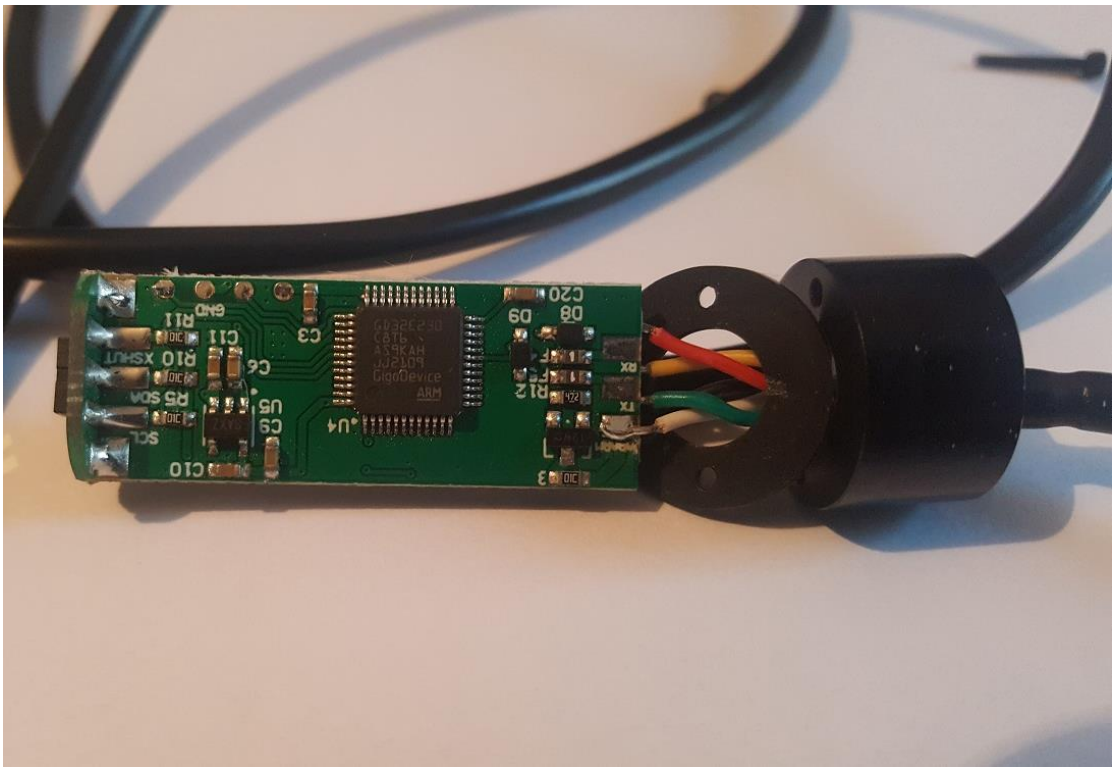


Abbildung 18: Bottom Seite mit RX und TX vorhanden

In der Zwischenzeit konnte die Stromaufnahme des Laser Distanz Sensors RS485 gemessen werden. Die Stromwerte bei 5V sind für Ruhe (kein Ansprechen durch die Software) 24mA und bei Betrieb (Ansprechen durch die Software) 25mA. Die Stromaufnahme kann gesenkt werden, indem man die Spannung erhöht, was aber für das Projekt keinen Sinn machen würde. Es müsste nur für den Sensor eine separate Speisung eingeplant werden, dass der Stromverbrauch reduziert werden kann.
Dies hätte zur Folge das ein Solarbetrieb erschwert wird.

Ein paar Tage später wurde der neue Sensor geliefert und dieser wurde vor Inbetriebnahme auseinandergeschraubt. Dieses Mal war es der Sensor welcher RX und TX verbunden hatte.

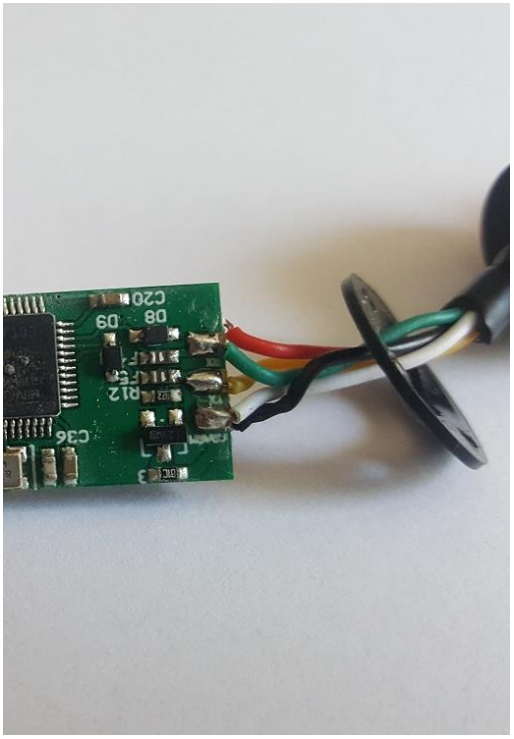


Abbildung 19: Grossaufnahme Sensor UART

Auch hier wurden die Stromwerte bei 5V gemessen.
Die Stromwerte bei 5V sind für Ruhe (kein Ansprechen durch die Software) 35mA und bei Betrieb (Ansprechen durch die Software) 36mA.
Aufgrund der hohen Werte ist der Sensor nicht für das Projekt geeignet.
Der vorprogrammierte Arduino wurde genommen und der Sensor angeschlossen.
Mit dem richtigen Sensor funktionierte der Code.

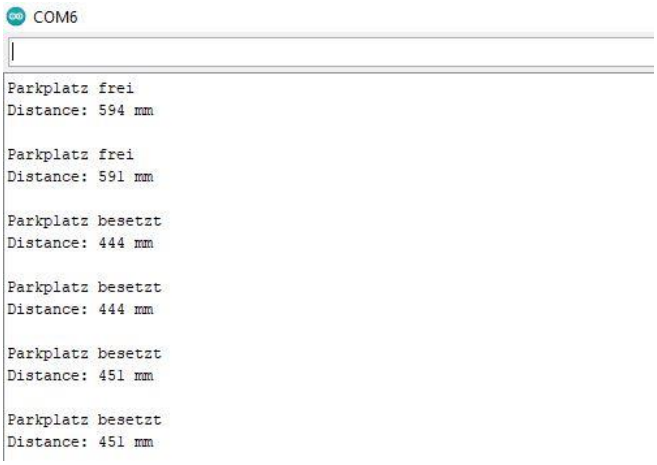


Abbildung 20: Ausgabe im Serial Monitor

Da es in den Unterlagen zu diesem Sensor Hinweise gibt, dass dieser Sensor Probleme haben könnte mit schwarzen Objekten, wurde dies mit einem Versuch bei einem schwarzen Fahrzeug in der Garage durchgeführt.
Das Problem hat sich zuerst nicht bestätigt, doch sobald das schwarze Fahrzeug gereinigt und ins Sonnenlicht gestellt wurde, konnte der Sensor keine richtigen Daten mehr empfangen.
Dies liegt vermutlich daran, dass die Farbe schwarz das Laserlicht schluckt und die Reflexionen zu gering für die Funktionsweise des Lasers sind.

Abbildung 21: Ausgabe im Serial Monitor schwarzes Fahrzeug

In diesem Moment war klar, dass dieser Sensor für das Projekt wegfallen wird.
Ausserdem zeigte dieser Versuch deutlich, dass Ultraschallsensoren besser für diese Anwendung geeignet sind, da diese von reflektiertem Licht nicht beeinflusst werden.

Testversuch mit Ultraschallsensor JSN-SR04T-V3.0

Der letzte Sensor ist nochmals ein Ultraschallsensor, welcher über die digitalen Ein- und Ausgänge eines Mikrokontrollers gesteuert werden kann.

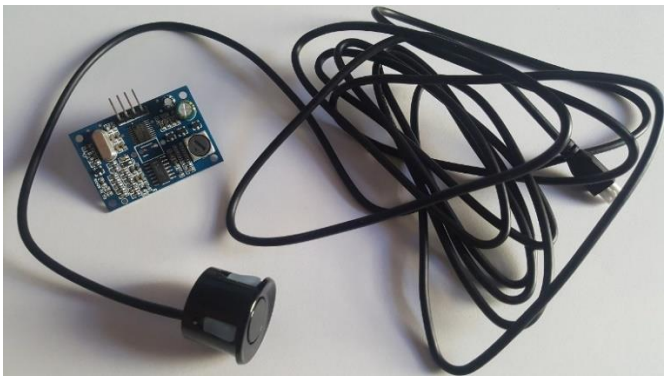


Abbildung 22: Ultraschallsensor JSN-SR04T-V3.0

Dieser Sensor kann bei 3.3V und 5V betrieben werden. Dieser Sensor hat den Vorteil, dass über einen zusätzlichen Widerstand diverse Modes eingestellt werden können. Auch bei diesem Sensor muss ein Gehäuse gefertigt werden, um ihn befestigen zu können. Es wird extra die Version 3.0 genannt, weil die älteren Versionen andere Möglichkeiten haben.

Mit einem Versuchsaufbau wird der Sensor getestet.
Folgendes Equipment wird dafür verwendet.

Equipment

1x JSN-SR04T-V3.0 Ultraschallsensor wasserfest von DFRobot
1x ESP32-S2-Saola-1R Developmentboard von ESPRESSIF
1x Laptop mit Arduino IDE
1x Fluke 179 Multimeter als Amperemeter

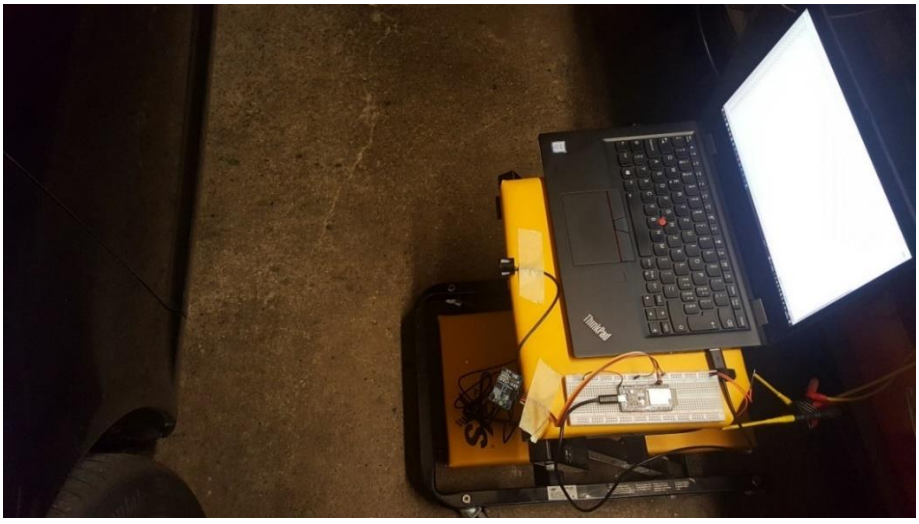


Abbildung 23: Messaufbau mit ESP32 und Sensor

Testversuch

Die Arduino IDE musste um die ESP32 Developmentboardpalette erweitert werden.
Der Softwarecode für den ESP32, welcher auf der Homepage von dfrobot.com verfügbar ist, bearbeitet, für die Testzwecke erweitert und auf den ESP32 geladen.
Der Sensor wird mit dem ESP32 verbunden und getestet. Die Stromaufnahme wird bei 3.3V und bei 5V durchgeführt.

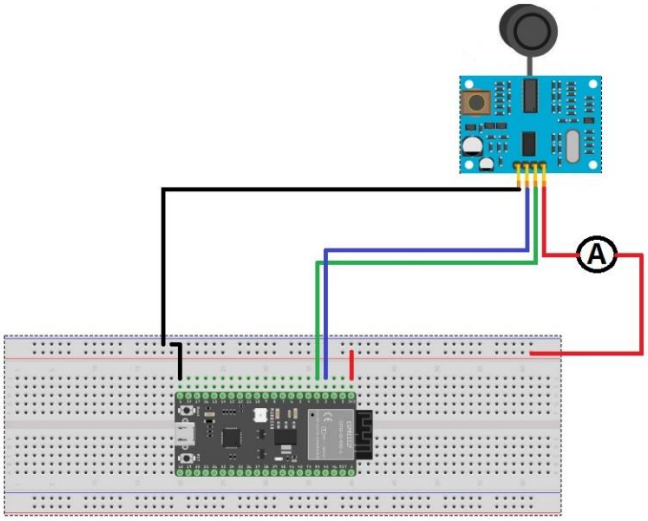


Abbildung 24: Schema ESP32 und Sensor bei 3.3V

Ergebnis

Der Ultraschall Sensor erfüllt das gewünschte Ziel und detektiert den Abstand. Dieser Sensor macht ab ca. 20cm von ihm entfernt keinen Unterschied mehr ob ein Objekt 1cm oder 20cm entfernt ist.
Die Software soll ab einem Abstand kleiner gleich 50cm den Parkplatz als besetzt melden.

Abbildung 25: Ausgabe auf dem Serial Monitor

Die Stromwerte des Sensors sind bei 3.3V für Ruhe (kein Ansprechen durch die Software) 3.5mA und bei Betrieb (Ansprechen durch die Software) 3.7mA.
Die Stromwerte bei 5V sind für Ruhe 4.5mA und bei Betrieb 4.7mA.

Diese Werte sprechen für diesen Sensor.

Mit einer kleinen Änderung auf dem Sensorboard können noch bessere Stromwerte erzielt werden. In der Dokumentation dieses Sensors steht, dass wenn ein 360kOhm Widerstand bei Mode eingelötet wird, dieser Sensor noch weniger Strom auf nimmt. Dieser Versuch wurde gemacht und das Sensorboard modifiziert.

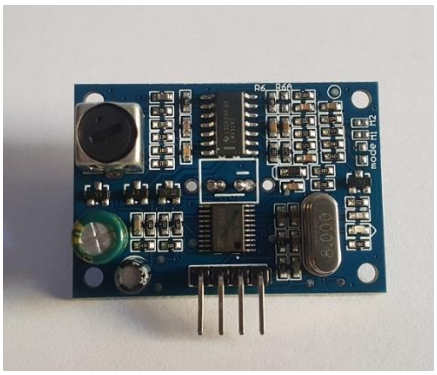


Abbildung 26: Sensorboard ohne Modifikation

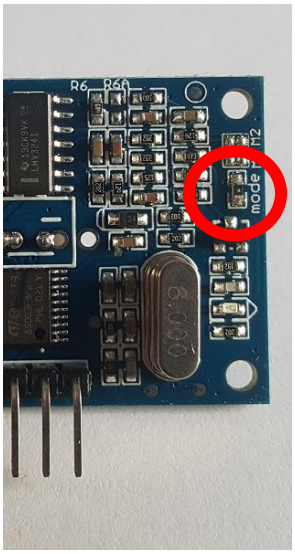


Abbildung 27: Sensorboard mit Modifikation

Die Stromwerte haben sich durch diese Änderung bei 3.3V für Ruhe (kein Ansprechen durch die Software) auf 0.45mA und bei Betrieb (Ansprechen durch die Software) auf 0.6mA geändert.
Gleich verhält es sich für die Stromwerte bei 5V für Ruhe auf 0.7mA und bei Betrieb auf 1.37mA.

Entscheidung zu den Sensoren

Alle drei Sensoren sind getestet und eine Entscheidung wurde getroffen.

Der Ultraschallsensor A02YYUW sowie der Laser Distanz Sensor fallen weg. Einerseits gab es diverse Probleme mit diesen beiden Sensoren, sei es die Kommunikation oder auch ihre Eigenschaften auf Farbe oder die Stromaufnahme, welche für dieses Projekt zu hoch sind.
Über den Preis für diese beiden Sensoren wurde noch nichts geschrieben, dies wird nun nachgeholt, da auch der Preis nicht für diese beiden Sensoren spricht.

Sensortyp	Preis	Stromaufnahme	Besonderes
Ultraschallsensor A02YYUW	25.90 CHF	**3.3V:** 5.25 bis 5.42mA **5V:** 6.94 bis 7.08mA	Probleme mit anderen Sensoren mit 9600 Bit/s Baudrate
Laser Distanz Sensor SEN0491 oder SEN0492	27.90 CHF	**SEN0492 (RS485) bei 5V:** 24 bis 25mA **SEN0491 (UART) bei 5V:** 35 bis 36mA	Probleme mit schwarzen Fahrzeugen
Ultraschallsensor JSN-SR04T-V3.0	19.90 CHF	**3.3V:** 3.5 bis 3.7mA **5V:** 4.5 bis 4.7mA **Modifikation mit 360kOhm Widerstand** **3.3V:** 0.45 bis 0.6mA **5V:** 0.7 bis 1.37mA	Verschiedene Modis möglich durch Widerstände

Tabelle 4: Sensortypen

Beim Laser Distanz Sensor ist man sich nicht sicher welchen Typen man erhält. Um zu sehen welchen Typ des Sensors man erhalten hat muss dieser auseinander gebaut werden.

Somit fällt die Entscheidung auf den Ultraschallsensor JSN-SR04T-V3.0.
Dieser kostet ein Stück weniger und reicht aus für dieses Projekt.
Der Sensor hat durch seine Modifikationsmöglichkeiten noch einen geringeren Stromverbrauch.
Sechs dieser Sensoren werden bestellt und das Projekt mit diesen Sensoren realisiert.

Mikrokontroller

Auswahl des Mikrokontrollers

Für das Projekt wird ein Mikrokontroller gebraucht, welcher die ganzen Sensoren steuert und die Daten an den Raspberry Pi senden kann. Durch die Versuche mit den Sensoren, konnten bereits erste Erfahrungen mit den verschiedenen Mikrokontrollern gemacht werden.

Der Mikrokontroller muss zwingend mindestens 12 nutzbare Ein- und Ausgänge haben und WLAN fähig sein.
Von Vorteil wäre es, wenn noch zusätzliche Ein- und Ausgänge über einen I/O Expander erweitert werden können oder ein Display über I2C angehängt werden könnte.

Die Auswahl des Mikrokontrollers findet zwischen einem Ardunio Mega 2560, einem ESP8266 mit Base Board Protoshield mit Spannungsregler und einem ESP32 S2 Saola 1R Developmentboard statt.

Mit einem angepassten Softwarecode für den Ultraschallsensor JSN-SR04T-V3.0 werden die drei Mikrokontroller programmiert und bei allen der Ultraschallsensor angeschlossen. Anschliessend wird die Stromaufnahme der Mikrokontroller ermittelt.

Testversuch mit dem Arduino Mega 2560

Der erste Mikrokontroller ist der Arduino Mega 2560. Dieser verfügt über keinen WLAN fähigen Prozessor.
Er weisst genügend nutzbare Ein- und Ausgänge auf. Dieser Mikrokontroller wurde im fünften Semester an der TEKO Zürich in den Fächern Digitaltechnik sowie Mikrokontrollertechnik verwendet. Dadurch konnten viele Erfahrungen gesammelt werden.

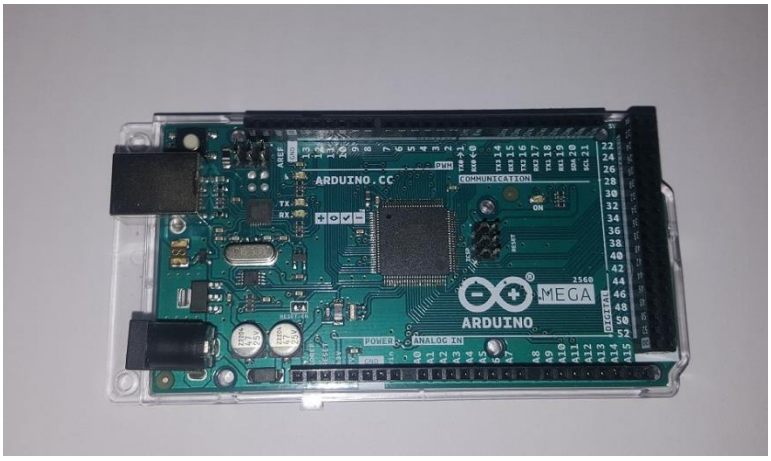


Abbildung 28: Arduino Mega 2560

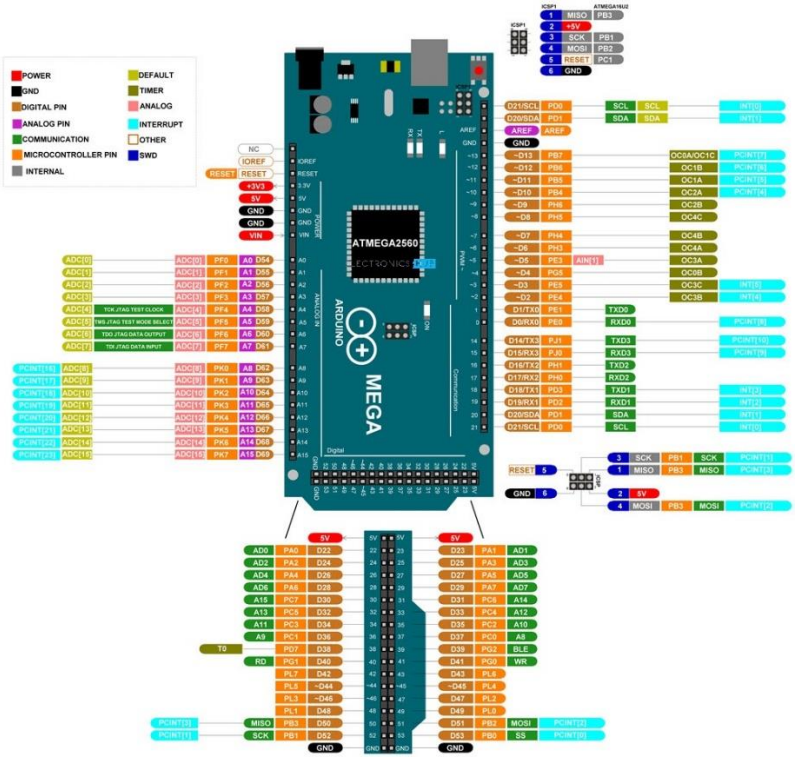


Abbildung 29: Arduino Mega 2560 Pinout

Mit einem Versuchsaufbau wird die Stromaufnahme des Mikrokontrollers getestet.
Folgendes Equipment wird dafür verwendet.

Equipment und Messaufbau

1x JSN-SR04T-V3.0 Ultraschallsensor wasserfest von DFRobot
1x Arduino Mega 2560 von Arduino
1x Laptop mit Arduino IDE
1x UM34C USB Tester Messgerät QC3.0 als Amperemeter von Hangzhou Ruideng Technology

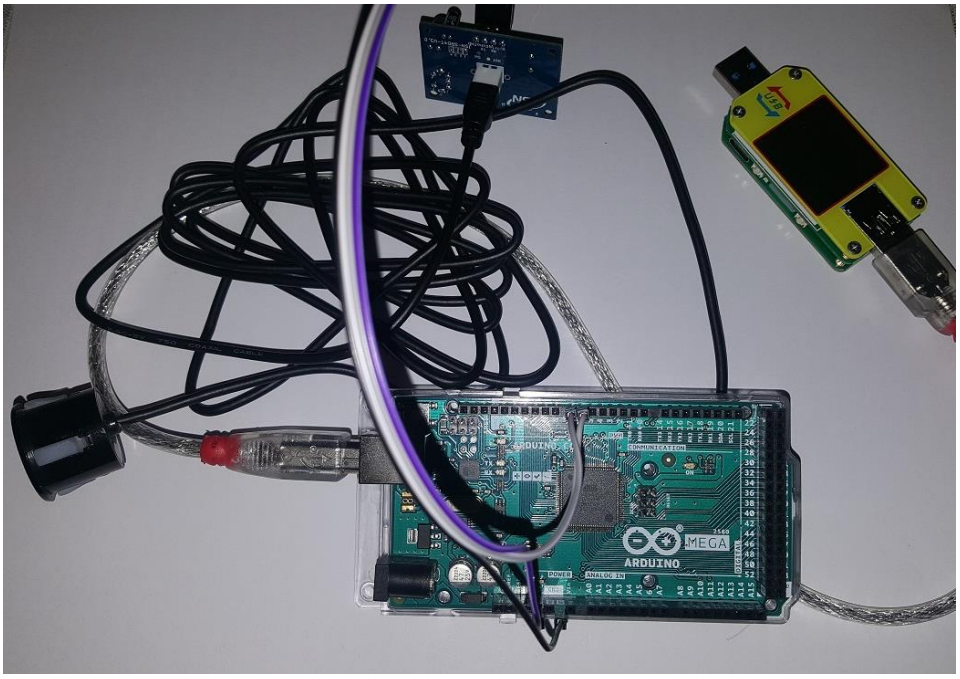


Abbildung 30: Messaufbau mit Arduino Mega 2560 und Sensor

Ergebnis

Die Stromwerte des Arduino Mega 2560 sind bei 3.3V für Ruhe (kein Ansprechen durch die Software) 79mA und bei Betrieb (Ansprechen durch die Software) 80mA.
Die Stromwerte bei 5V sind für Ruhe 79mA und bei Betrieb 81mA.
Der angeschlossene Sensor mit der Modifikation des 360kOhm Widerstandes beeinflusst mit seiner Stromaufnahme die Messung bis maximal 1.37mA.

Testversuch mit dem ESP8266

Der zweite Mikrokontroller ist der ESP8266. Er verfügt über einen WLAN fähigen Prozessor. Dieser Mikrokontroller wurde im fünften Semester an der TEKO Zürich im Fach Mikrokontrollertechnik verwendet und wurde für die Semesterarbeit verwendet. Dadurch konnten viele Erfahrungen gesammelt werden.

Abbildung 31: ESP8266 mit Base Board Protoshield mit Spannungsregler

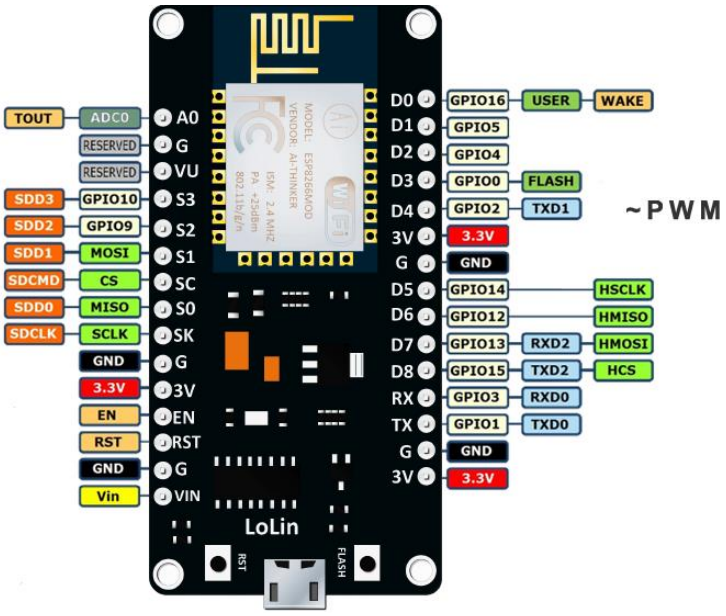


Abbildung 32: ESP8266 Pinout

Wichtig zum Wissen für diesen Mikrokontroller ist, dass nicht alle GPIO Anschlüsse effektiv für alle Anwendungen genutzt werden können. Dies ist in der Tabelle von der Internetseite https://randomnerdtutorials.com/esp8266-pinout-reference-gpios zu entnehmen.

Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Abbildung 33: Tabelle mit den nutzbaren GPIO's

Mit einem Versuchsaufbau wird die Stromaufnahme des Mikrokontrollers getestet.
Folgendes Equipment wird dafür verwendet.

Equipment und Messaufbau

1x JSN-SR04T-V3.0 Ultraschallsensor wasserfest von DFRobot
1x ESP8266 NodeMCU V3 von ESPRESSIF
1x ESP8266 NodeMCU V3 Base Board Protoshield mit Spannungsregler von Bastelgarage.ch
1x Laptop mit Arduino IDE
1x UM34C USB Tester Messgerät QC3.0 als Amperemeter von Hangzhou Ruideng Technology

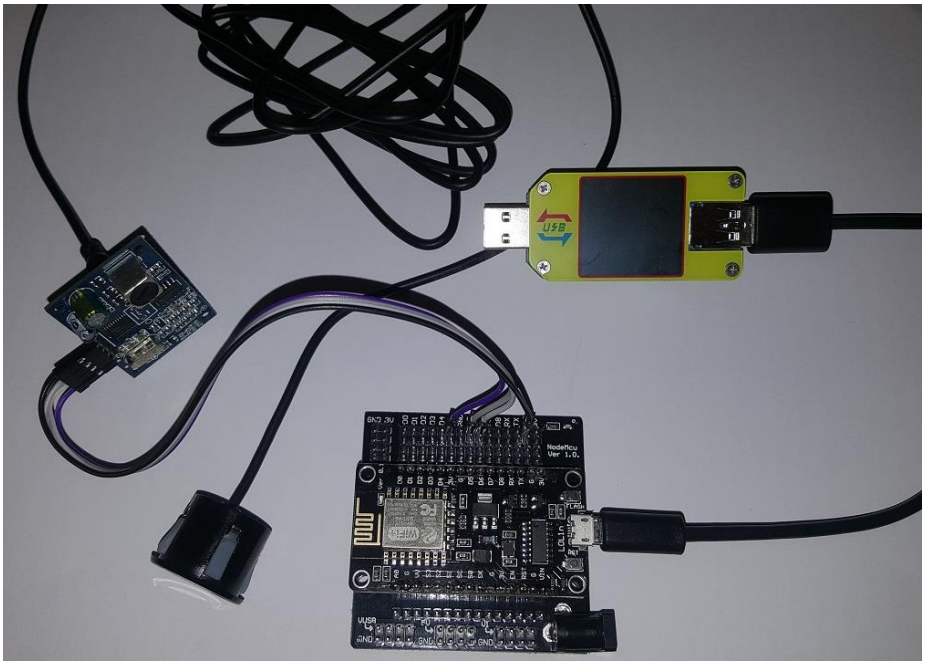


Abbildung 34: Messaufbau mit ESP8266 und Sensor

Ergebnis

Die Stromwerte des ESP8266 sind bei 3.3V für Ruhe (kein Ansprechen durch die Software) 318mA und bei Betrieb (Ansprechen durch die Software) 330mA.
Die Stromwerte bei 5V sind für Ruhe 318mA und bei Betrieb 331mA.
Der angeschlossene Sensor mit der Modifikation des 360kOhm Widerstandes beeinflusst mit seiner Stromaufnahme die Messung bis maximal 1.37mA.

Testversuch mit dem ESP32 S2 Saola 1R Developmentboard

Der dritte Mikrokontroller ist das ESP32 S2 Saola 1R Developmentboard. Dieser verfügt über keinen WLAN fähigen Prozessor.
Er weisst genügend nutzbare Ein- und Ausgänge auf. Dieser Mikrokontroller wurde im fünften Semester an der TEKO Zürich in den Fächern Digitaltechnik sowie Mikrokontrollertechnik verwendet. Dadurch konnten viele Erfahrungen gesammelt werden.

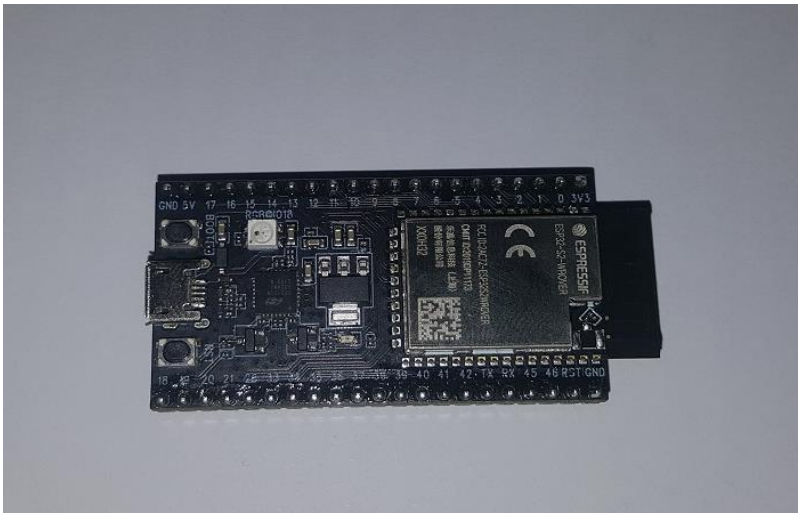


Abbildung 35: ESP32 S2 Saola 1R

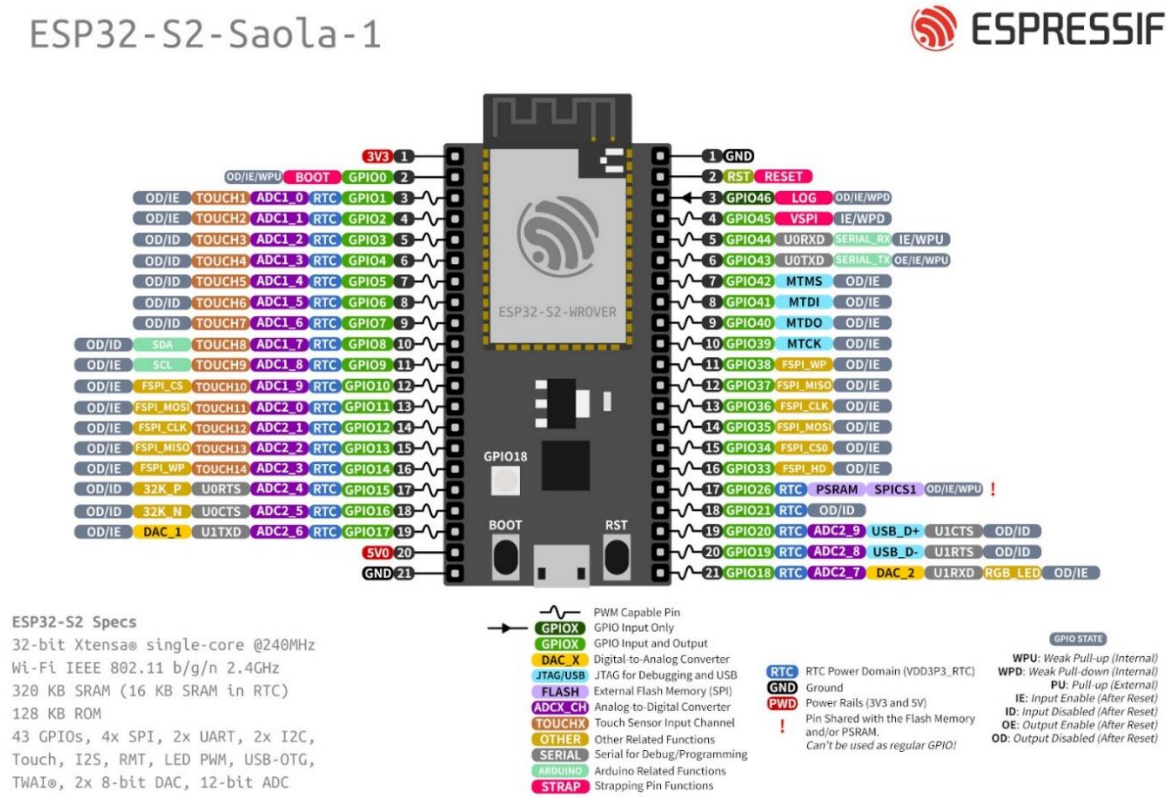


Abbildung 36: ESP32 S2 Saola 1 Pinout

Mit einem Versuchsaufbau wird die Stromaufnahme des Mikrokontrollers getestet.
Folgendes Equipment wird dafür verwendet.

Equipment und Messaufbau

1x JSN-SR04T-V3.0 Ultraschallsensor wasserfest von DFRobot
1x ESP32-S2-Saola-1R Developmentboard von ESPRESSIF
1x Laptop mit Arduino IDE
1x UM34C USB Tester Messgerät QC3.0 als Amperemeter von Hangzhou Ruideng Technology

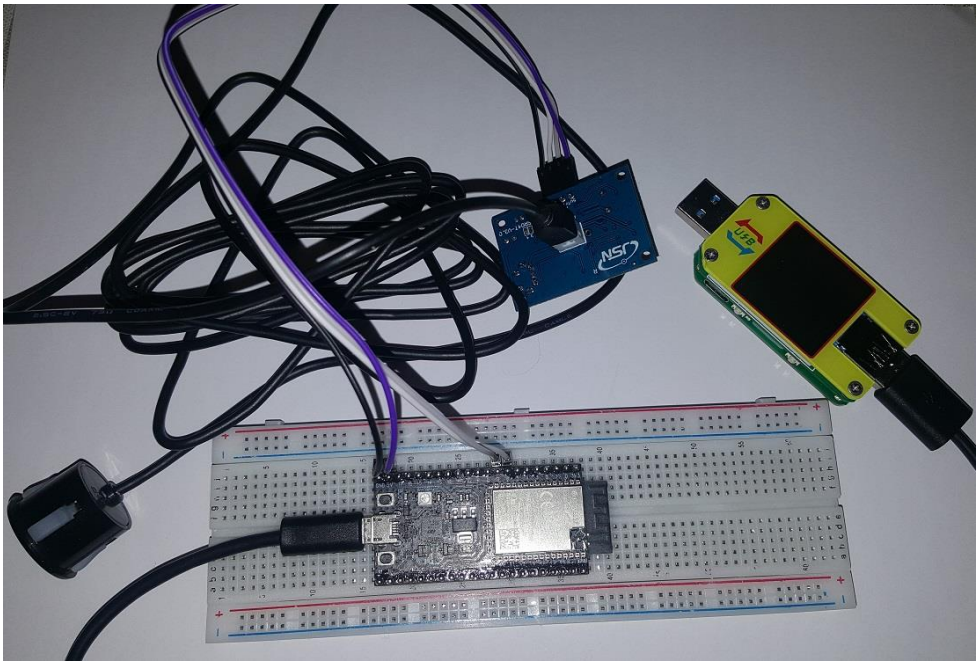


Abbildung 37: Messaufbau mit ESP32 S2 Saola 1R und Sensor

Ergebnis

Die Stromwerte des ESP32 S2 Saola 1R sind bei 3.3V für Ruhe (kein Ansprechen durch die Software) 31mA und bei Betrieb (Ansprechen durch die Software) 31mA.
Die Stromwerte bei 5V sind für Ruhe 31mA und bei Betrieb 32mA.
Der angeschlossene Sensor mit der Modifikation des 360kOhm Widerstandes beeinflusst mit seiner Stromaufnahme die Messung bis maximal 1.37mA.

Entscheidung zu den Mikrokontrollern

Alle drei Mikrokontroller sind getestet und eine Entscheidung wurde getroffen.

Der Arduino Mega 2560 und der ESP8266 fallen weg. Der Arduino Mega 2560 hat kein WLAN und der ESP8266 nimmt am meisten Strom auf. Der ESP32 S2 Saola 1R ist der Einzige der drei Mikrokontroller, welcher alle Anforderungen erfüllt.
Über den Preis für dieser Mikrokontroller wurde noch nichts geschrieben, dies wird nun nachge holt.

Mikrokontrollertyp	Preis	Stromaufnahme	Besonderes
Arduino Mega 2560	46.80 CHF	79 bis 81mA	Viele I/O's Kein WLAN Hoher Preis
ESP8266	9.50 CHF	318 bis 331mA	Wenige I/O's WLAN Hohe Stromaufnahme
ESP32 S2 Saola 1R	8.91 CHF	31 bis 32mA	Viele I/O's WLAN Geringe Stromaufnahme Geringer Preis

Tabelle 5: Mikrokontrollertypen

Somit fällt die Entscheidung auf den ESP32 S2 Saola 1R.
Dieser ist am günstigsten und erfüllt alle Anforderungen für dieses Projekt.
Das Projekt wird mit diesem Mikrokontroller realisiert.

Raspberry Pi

Die Entscheidung für dieses Projekt einen Raspberry Pi zu benutzen war schnell evaluiert.
Im 5. Semester an der TEKO Zürich konnte im Fach Mikrokontroller, infolge einer Semesterarbeit, ein Raspberry Pi für ein Projekt genutzt werden. Durch den Mosquitto MQTT-Broker, welcher auf dem Raspberry Pi installiert wurde, konnte zwischen einem Mikrokontroller und dem Raspberry Pi über WLAN kommuniziert werden und die Sensorwerte, welche dem Mikrokontroller übe rgeben wurden, an den Raspberry Pi gesendet werden. Dieser konnte dann durch die Installation von Node-RED auf einem User Interface die Daten grafisch anzeigen.
Aufgrund von Corona und der Bauteilknappheit war es schwierig an einen Raspberry Pi heranzukommen.
Es gab aber noch eine kleine Anzahl von Bundles, welche ein wenig mehr Inhalt haben als nur den Raspberry Pi.
So wurde von der Firma Okdo ein Bundle mit einem Raspberry Pi 4 mit 4GB bestellt.
Dieses Bundle hat noch den Vorteil, dass es mit Kühlkörpern, Lüfter, Gehäuse und SD-Karte kommt, um direkt beginnen zu können und auch problemlos den Raspberry Pi über eine längere Zeit in Betrieb haben zu können.

Abbildung 38: Okdo Raspberry Pi 4 Bundle

Mit einem Raspberry Pi 4 Model B, welcher mit dem WLAN verbunden ist und der Mosquitto Server sowie Node-RED installiert sind, wird die Stromaufnahme gemessen.

Equipment und Messaufbau

1x Raspberry Pi 4 Model B 4GB von Raspberry Pi Foundation
1x UM34C USB Tester Messgerät QC3.0 als Amperemeter von Hangzhou Ruideng Technology

Abbildung 39: Messaufbau mit Raspberry Pi 4

Ergebnis

Der Raspberry Pi kann ohne Lüfter betrieben werden. Dann braucht er zwischen 340 bis 400mA. Wird der Lüfter zur Kühlung verwendet kann dieser entweder an 3.3V oder 5V angeschlossen werden. Bei 3.3V braucht es dann 370 bis 450mA und bei 5V erreicht man zwischen 410 bis 480mA.

Bezeichnung	Preis	Stromaufnahme	Besonderes
Okdo Bundle mit Raspberry Pi 4 Model B 4GB	129.90 CHF	**Ohne Lüfter:** 340 bis 400mA **Lüfter 3.3V:** 370 bis 450mA **Lüfter 5V:** 410 bis 480mA	Mit dem Mosquitto MQTT-Broker und der Node-RED Software ideal für IoT Anwendungen

Tabelle 6: Raspberry Pi 4

I/O Erweiterung

Mit dem ESP32 kann dieses Projekt noch mit vielen Erweiterungen ausgestattet werden.
Um noch andere Sensoren oder sonstige zusätzliche Hardware anzuschliessen, braucht es aber immer freie Ein- und Ausgänge.
Auch wenn der ESP32 viele dieser Ein- und Ausgänge besitzt und durch das Projekt nicht alle benutzt werden, kann über eine Erweiterung in Form des PCF8574 I2C Portexpander der ESP32 über die Anschlüsse SDA und SCL zusätzlich mit einem Erweiterungsboard um 8 GPIO's erweitert werden. Der Vorteil dieser Erweiterungsboards ist, dass bis zu acht dieser Boards aneinandergesteckt werden können und mit den Jumpern auf dem Erweiterungsboard kann die Adresse für das I2C eingestellt werden. Somit wären 64 zusätzliche GPIO's vorhanden.
Da möglichst viele Ein- und Ausgänge auf dieses Board transferiert werden sollten, wird mit einem kleinen Messaufbau die Stromaufnahme dieses Boards gemessen.

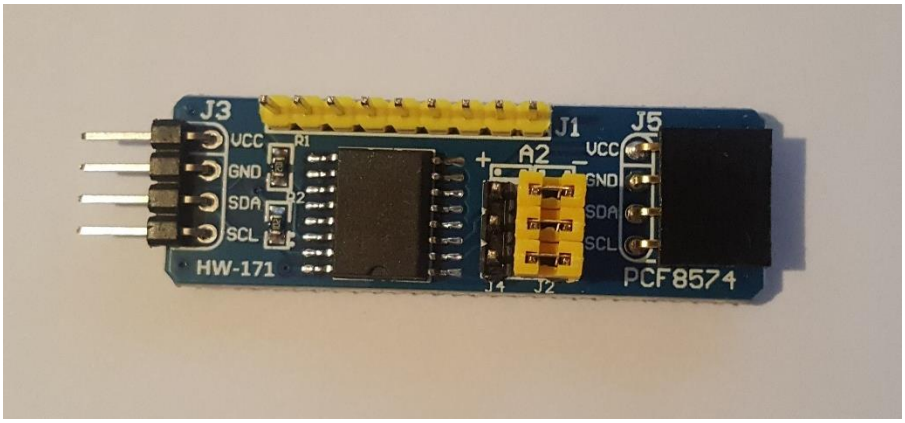


Abbildung 40: PCF8574 I2C Erweiterungsboard

Equipment und Messaufbau

1x PCF8574 I2C Portexpander I/O Erweiterung 8 Pin von Bastelgarage.ch
1x 1x Fluke 179 Multimeter als Amperemeter
1x 24V Netzgerät für 3.3V und 5V

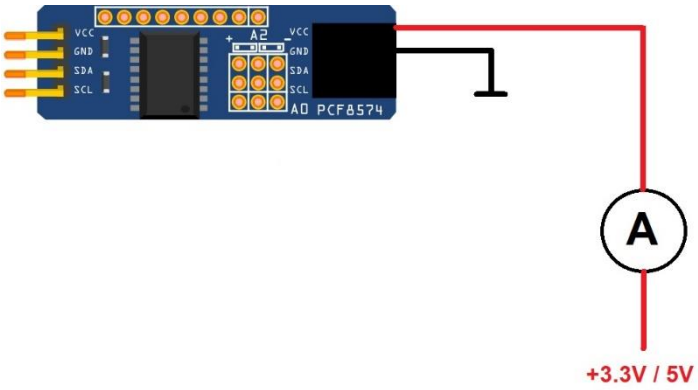


Abbildung 41: Messaufbau PCF8574

Ergebnis und Entscheid

Die Entscheidung dieses Erweiterungsboard für das Projekt einzusetzen ist als sinnvoll anzunehmen, um möglichst viele Ein- und Ausgänge des ESP32 freizuhalten. Bei 3.3V ist die Stromaufnahme 3.36mA und bei 5V sind es 5.05mA. Jeder Pin dieses Erweiterungsboards kann 25mA schalten.

Bezeichnung	Preis	Stromaufnahme	Besonderes
PCF8574 I2C Portexpander I/O Erweiterung 8 Pin	5.90 CHF	**3.3V:** 3.36mA **5V:** 5.05mA	Bis zu acht dieser Boards können aneinander gesteckt werden.

Tabelle 7: PCF8574 I/O Erweiterung

Temperatursensor

Der ESP32 sowie diverse Zusatzelektronik soll in ein wasserdichtes Gehäuse verbaut werden.
Um die Temperatur in diesem Gehäuse überwachen zu können, braucht es einen Temperatursensor.
Durch den Temperatursensor soll eine Überhitzung der Elektronik im Gehäuse verhindert werden.
Der Temperatursensor soll anhand der Software auf dem ESP32 eine automatische Lüftersteuerung einschalten, wenn ein Schwellwert erreicht ist. Hierbei soll geschaut werden, dass der Temperatursensor auch Temperaturen messen kann, im Bereich -30°C bis +80°C.

Die Entscheidung fällt auf den DHT22 Temperatur und Luftfeuchtigkeitssensor. Dieser erfüllt die gewünschten Anforderungen.
Da es für dieses Projekt wichtig ist, von allen Komponenten die Stromaufnahme zu wissen, wird auch mit diesem Sensor ein kleiner Messaufbau gemacht.

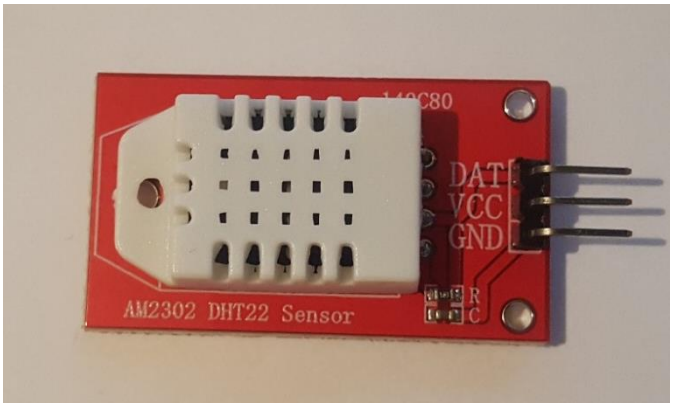


Abbildung 42: Temperatursensor AM2302 DHT22

Equipment und Messaufbau

1x DHT22 Temperatur und Luftfeuchtigkeitssensor von Bastelgarage.ch
1x Fluke 179 Multimeter als Amperemeter
1x 24V Netzgerät für 3.3V und 5V

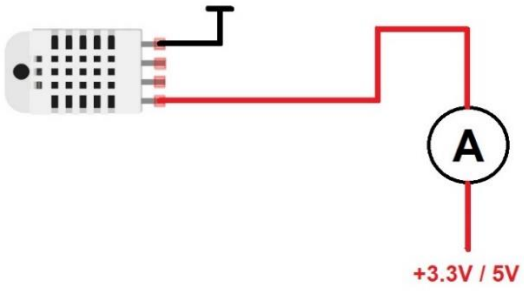


Abbildung 43: Messaufbau Temperatursensor

Ergebnis und Entscheid

Dieser Temperatursensor erfüllt die gewünschten Anforderungen und wird für das Projekt eingesetzt. Bei 3.3V ist die Stromaufnahme 0.17mA und bei 5V sind es 0.23mA.

Bezeichnung	Preis	Stromaufnahme	Besonderes
DHT22 AM2302 Temperatur und Luftfeuchtigkeitssensor	11.50 CHF	**3.3V:** 0.17mA **5V:** 0.23mA	Genauigkeit +/- 0.5°C

Tabelle 8: Temperatursensor

Ventilator

Um eine Überhitzung des ESP32 zu verhindern, sollen zwei Ventilatoren eingesetzt werden. Diese Ventilatoren sollen, wenn ein Grenzwert im Gehäuse überschritten wird, automatisch einschalten. Die Ventilatoren sollen so betrieben werden, dass einer saugend und der andere blasend verbaut ist. Wichtig ist, dass diese Ventilatoren genügend Luft bewegen können, um schnell die Temperatur im Gehäuse runterzukühlen. So soll der Stromverbrauch nicht ausser Acht gelassen werden, da dieses System solarbetrieben funktionieren soll. Da die meisten Ventilatoren mit 5V Spannung betrieben werden, muss bedacht werden, dass die Signale vom ESP32 nur 3.3V schalten können. So muss dazwischen ein Relais eingebaut werden. Ein 5V Relais kann auch von einem 3.3V Ausgang des ESP32 geschalten werden.
Es braucht auch hier einen Testversuch, um zu evaluieren wie viel Strom gebraucht wird.
Es gibt im Funduino-Kit der TEKO Zürich ein Relais. Ein baugleiches Relais wurde bestellt und wird benutzt, um die Ventilatoren zu testen. Der Entscheid fiel auf die Ventilatoren der Marke Noctua, diese haben eine hohe Laufruhe und sind sehr leise. Sie können 8.20m^3/h bewegen.

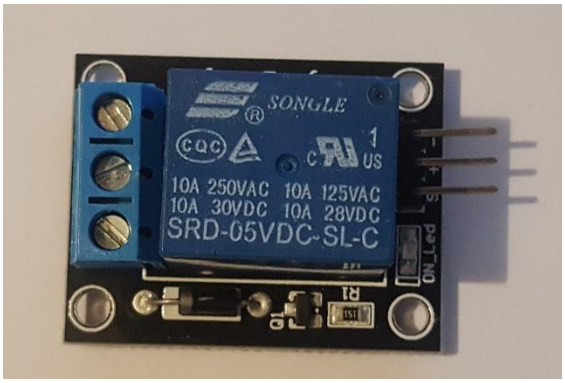


Abbildung 44: Relais SRD-05VDC

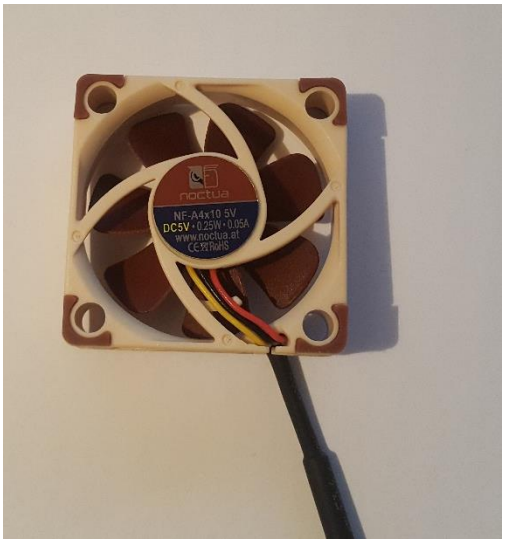


Abbildung 45: Noctua 5V Ventilator

Equipment und Messaufbau

2x NF-A4x10 5V Lüfter von Noctua
1x Single Relais Board 5V von Parallax
1x Fluke 179 Multimeter als Amperemeter
1x 24V Netzgerät für 3.3V und 5V

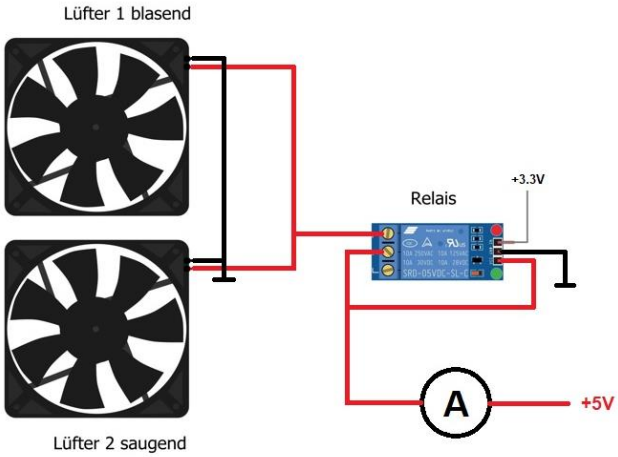


Abbildung 46: Messaufbau Relais und Ventilatoren

Ergebnis und Entscheid

Die Ventilatoren brauchen jeweils 50mA bei 5V. Das Relais braucht ca. 35mA, um durchzuschalten. Da auf dem Relaisboard noch eine LED verbaut ist, braucht diese auch noch rund 15mA.
In diesem Fall braucht es zwei Pins vom I/O Erweiterungsboard, um das Relais mit genügend Strom durchzuschalten. Es wäre möglich die Ventilatoren auch bei 3.3V zu betreiben. Dann wäre die Stromaufnahme pro Ventilator 30mA und die Luftbewegung nimmt minim ab. Auch in diesem Fall braucht es ein Relais, welches die Ventilatoren schalten muss.
Es wurde entschieden, beide Ventilatoren mit 3.3V über das Relais zu betreiben, um möglichst viel Strom sparen zu können.

Bezeichnung	Preis	Stromaufnahme	Besonderes
NF-A4x10 5V Lüfter	20.90 CHF	**3.3V:** 30mA **5V:** 50mA	Leise, hohe Langlebigkeit, hohe Luftbewegung
Single Relais Board 5V	9.89 CHF	**5V:** 50mA	

Tabelle 9: Ventilator und Relais

Solar

Durch alle Testaufbauten konnten die Stromwerte der einzelnen Komponenten ermittelt werden. Dies ist wichtig, um ein Solarpanel zu wählen, welches genug Strom liefern kann, um den Akku zu laden. Diese Werte sind in untenstehender Tabelle nochmals aufgelistet und zusammengerechnet. Aufgrund eines kleinen Tests mit dem ESP32 und einem der Ultraschallsensoren, konnte gemessen werden, dass wenn der ESP32 über WLAN kommuniziert, seine Stromaufnahme auf 80mA steigt. Diese Erkenntnis muss zwingend in diese Tabelle aufgenommen werden.

Anzahl der Komponenten	Bezeichnung	Stromaufnahme	Strom total
6	**Ultraschallsensor JSN-SR04T-V3.0**	**3.3V:** 0.45 bis 0.6mA	**3.6mA**
1	**ESP32 S2 Saola 1R**	**5V mit WLAN:** 80mA	**80mA**
1	**PCF8574 I2C Portexpander I/O Erweiterung 8 Pin**	**3.3V:** 3.36mA	**3.36mA**
1	**DHT22 AM2302 Temperatur und Luftfeuchtigkeitssensor**	**3.3V:** 0.17mA	**0.17mA**
2	**NF-A4x10 5V Lüfter**	**3.3V:** 30mA	**60mA**
1	**Single Relais Board 5V**	**5V:** 50mA	**50mA**
Stromaufnahme total: 197.13mA			

Tabelle 10: Stromaufnahme gesamt

Wichtig für die Auswahl eines Solarpanels und des Akkus ist, dass die Stromaufnahme von knapp 200mA nur bei einem heissen Tag zu Stande kommt, wenn die Temperatur über 50°C im Gehäuse steigen kann. Dieser Fall wird nur im Hochsommer vorkommen können und dann scheint die Sonne dementsprechend auch genug um den Akku wieder zu laden. Es kann daher angenommen werden, dass der alltägliche Gebrauch des Systems um die 90mA sein wird. Die 90mA setzen sich aus den sechs Ultraschallsensoren, dem ESP32, der I/O Erweiterung sowie dem Temperatursensor zusammen.
Da bisher noch keine Erfahrungen mit Solaranlagen gesammelt werden konnten, wird bei der Bastelgarage.ch ein Solarmanager bestellt und bei Mouser ein Solarpanel. Mit diesen beiden Komponenten wird ein Testversuch zusammengebaut.

Abbildung 47: Solarpanel

Abbildung 48: Solar Power Manager mit Akkus

Equipment und Messaufbau

1x Solarmodule und Solarzellen Colossal 6V 9W Solarpanel - 9.0 Watt von Adafruit
1x Solar Power Manager (C) für 6-24V Solarpanel von Waveshare
3x Li-Ion Akku 3.7V 3400mA NCR18650B 18650 mit Schutzelektronik von VariCore
1x Fluke 179 Multimeter als Amperemeter

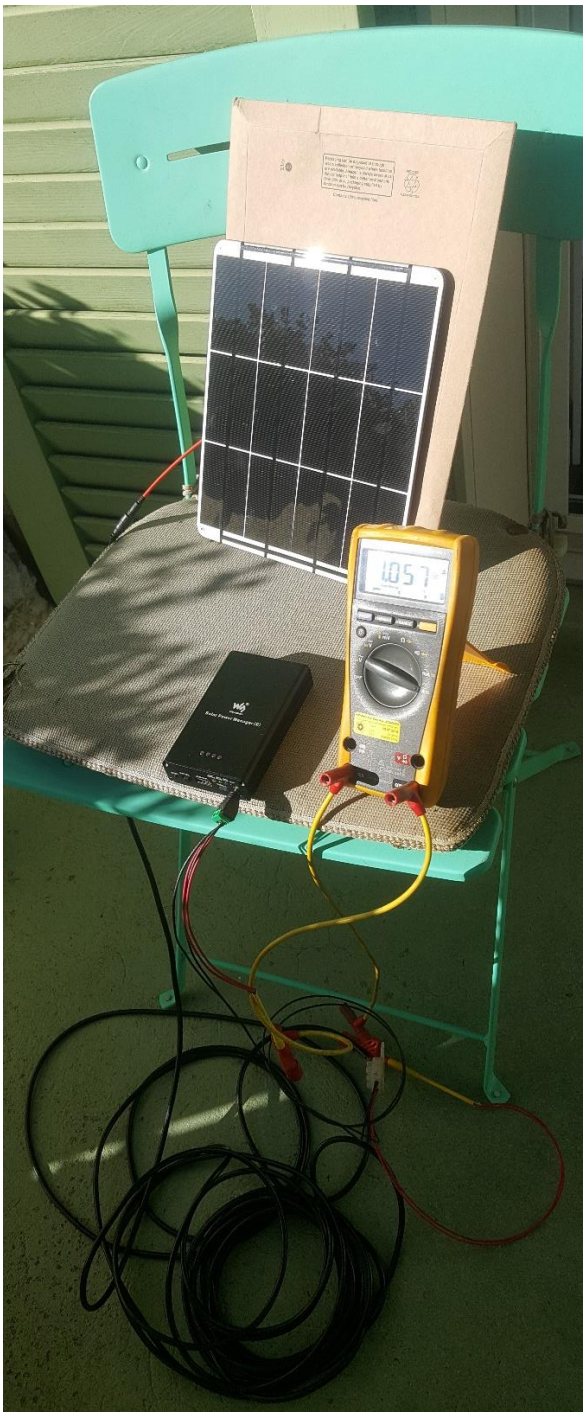


Abbildung 49: Messaufbau Solarpanel und Solar Power Manager

Ergebnis

Der Solar Power Manager kann drei der Li-Ionen Akkus aufnehmen und laden. Das Solarpanel kann direkt am Solar Power Manager angeschlossen werden. Bei optimalen Sonnenbedingungen konnten die Akkus mit 1.3A geladen werden. Alle 3 Akkus zusammen haben 10.2Ah. Das würde bedeuten, dass eine Stunde lang 10.2 Ampere gezogen werden können. Ebenfalls würde es mit den 1.3A, welche das Panel liefert, knapp acht Stunden Sonne brauchen, um die Akkus von leer auf voll zu laden. Acht Stunden Sonne wird nur in den Sommermonaten erreicht. Ausserdem muss die Sonne dann perfekt auf das Panel treffen, ansonsten nimmt die Stromgewinnung sehr schnell ab.
Geht man beim Projekt davon aus, dass der Normalzustand mit knapp 90mA mit den vollen Akkus von 10.2Ah betrieben werden können, kann die Anzahl Tage ohne laden berechnet werden.

Berechnung der Tage ohne nachladen: ((10.2Ah / 0.09A) / 24h) = 4.7 Tage

Dieser Wert wäre in Ordnung, jedoch kann es ab September sein, dass Nebel vorhanden ist am Standort der TEKO Zürich und somit würde dieser Wert eventuell nicht ausreichen. In den Sommermonaten besteht kein Bedenken, dass diese Akkuladung nicht ausreichen würde. Die Komponenten wären gut, jedoch so nicht optimal.
Für diese Anwendung muss Strom gespart werden und wenn Sonne vorhanden ist viel Strom zum Aufladen der Akkus zur Verfügung stehen.
Durch diesen Fall wird ein Soll-Ziel zu einem Muss-Ziel. Das Ziel Strom sparen durch Abschaltung von 21:00 abends bis 7:00 Uhr morgens muss stattfinden.
Ein Vorteil des ESP32 ist, dass dieser mehrere Schlafmodi besitzt, wie dem Datenblatt zu entnehmen ist.
Den wenigsten Verbrauch hat der ESP32 im Deep-Sleep Modus. Die Werte im Datenblatt wurden mit einem Testversuch überprüft. Es konnten nicht die Werte im Datenblatt gemessen werden, jedoch fiel die Stromaufnahme des ESP32 von 80mA auf 1 bis 2mA runter.
Somit ergibt sich eine neue Rechnung für die Tage ohne nachladen.

Berechnung der Tage ohne nachladen: (14h x 0.09A) + (10h x 0.008A) = 1.34Ah

(10.2Ah / 1.34Ah) = 7.6 Tage

Die Anpassung den ESP32 in den Deep-Sleep für 10 Stunden zu schicken, muss unbedingt beim Realisieren des Softwarecodes für den ESP32 bedacht werden.

Entscheid

Der Entscheid fällt auf die gewählten Komponenten. Das Solarpanel liefert für seine Grösse bei Sonneneinstrahlung ca. 1.5A Strom. Ausserdem ist der Platz, an dem dieses System stehen soll, recht begrenzt. Es muss klar sein, dass im Winter die Panels vom Schnee befreit werden müssen. Diese Panels haben keine Heizung drin, was auch nicht der Sinn wäre, da dies zusätzlich Strom verbrauchen würde. Um die Akkus schnell laden zu können fällt der Entscheid darauf, zwei dieser Solarpanel parallel zu schalten, um ca. 3A von den Panels beziehen zu können. Diese Panels werden genau gleich ausgerichtet, um keinen grossen Unterschied zwischen den Panels zu haben.
Der Solar Power Manager hat im Testversuch sich als ideal erwiesen. Sollte mal ein anderes Panel montiert werden, kann dieser Solar Power Manager auch Panels von 6-24V regeln. Dies ist über einen Dip-Switch einstellbar.

Bezeichnung	Preis	Stromabgabe	Besonderes
Solarmodule und Solarzellen Colossal 6V 9W Solarpanel - 9.0 Watt	2x 81.29 CHF	**6V:** je 1.5A	Anschlüsse im Panel, um an einen Rahmen zu befestigen, gute Stromabgabe, gemacht für aussen Anwendungen
Solar Power Manager (C) für 6-24V Solarpanel	23.90 CHF		Kann für Panel von 6-24V benutzt werden, kann über USB-C extern die Akkus laden, diverse LED's für Zustände
Li-Ion Akku 3.7V 3000mA NCR18650B 18650 mit Schutzelektronik	3x 9.90 CHF		Auch wenn hier mit 3000mA angegeben, auf der Herstellerseite mit 3400mA angegeben

Tabelle 11: Solar und Zubehör

Realisierung

Software des ESP32

Die Software ist das Herzstück dieses Projekts. Die Software auf dem ESP32 steuert die ganzen Sensoren, die Kommunikation vom und zum Raspberry Pi und auch alle Funktionen auf dem ESP32.
Ein Teil dieses Softwarekonstrukts konnte zum einem aus dem Projekt im 5. Semester an der TEKO Zürich übernommen werden, der andere Teil wird selbst aufgebaut und aus den Versuchsaufbauten eingebunden.
Der ganze Softwarecode wird in der Arduino DIE programmiert.
Für das ESP32 S2 Saola 1R Development Board mussten zusätzliche Boardverwalter-URL's eingebunden werden.

Für das ESP32 S2 Saola Board braucht es folgende Links:
https://dl.espressif.com/dl/package_esp32_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

Die Software wird in einzelnen Schritten realisiert. So können einzelne Teilsequenzen getestet und abgeschlossen werden.
Um die Software zu schreiben, muss ein kompletter Testaufbau aufgebaut werden. So können die einzelnen Pins des ESP32 auch in der Software richtig definiert werden.
Ein grosser Teil dieser Software ist auch Bestandteil für die Kommunikation ins Node-RED.
Die Software auf dem ESP32 muss die Sensordaten sowie diverse Befehle ins Node-RED senden können und auch Befehle empfangen können.

Es werden einzelne Teile des Softwarecodes erklärt, jedoch wird nicht der ganze Code hier in die Dokumentation kopiert. Der Code wird in seiner Endversion in den technischen Anhängen drin sein, sowie auf einem USB-Stick für eine Weiterführung des Projektes oder falls es durch die TEKO Zürich noch Anpassungen geben sollte.

Testaufbau für die Software des ESP32

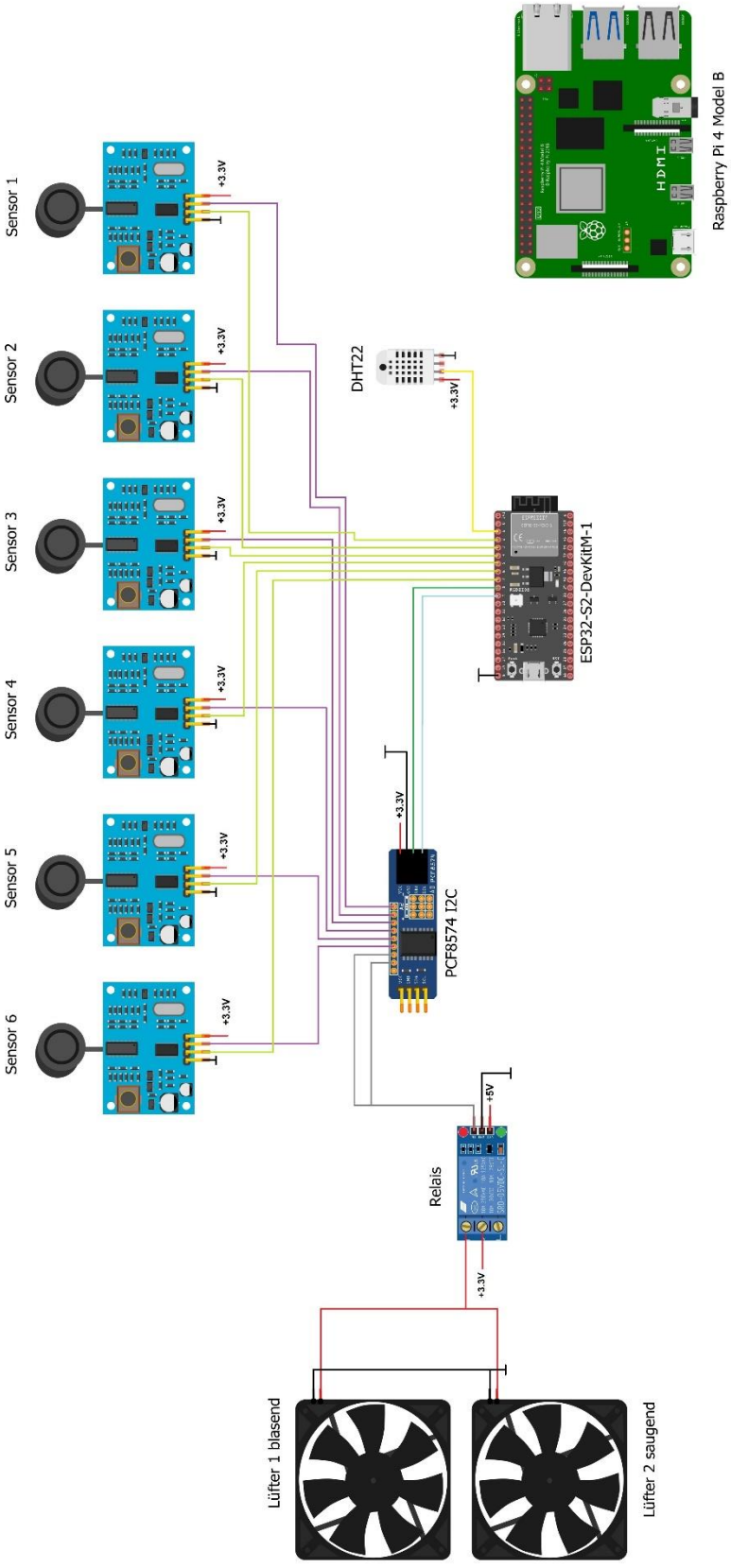


Abbildung 50: Testaufbau für die ESP32 Software

Flussdiagramm der Software des ESP32

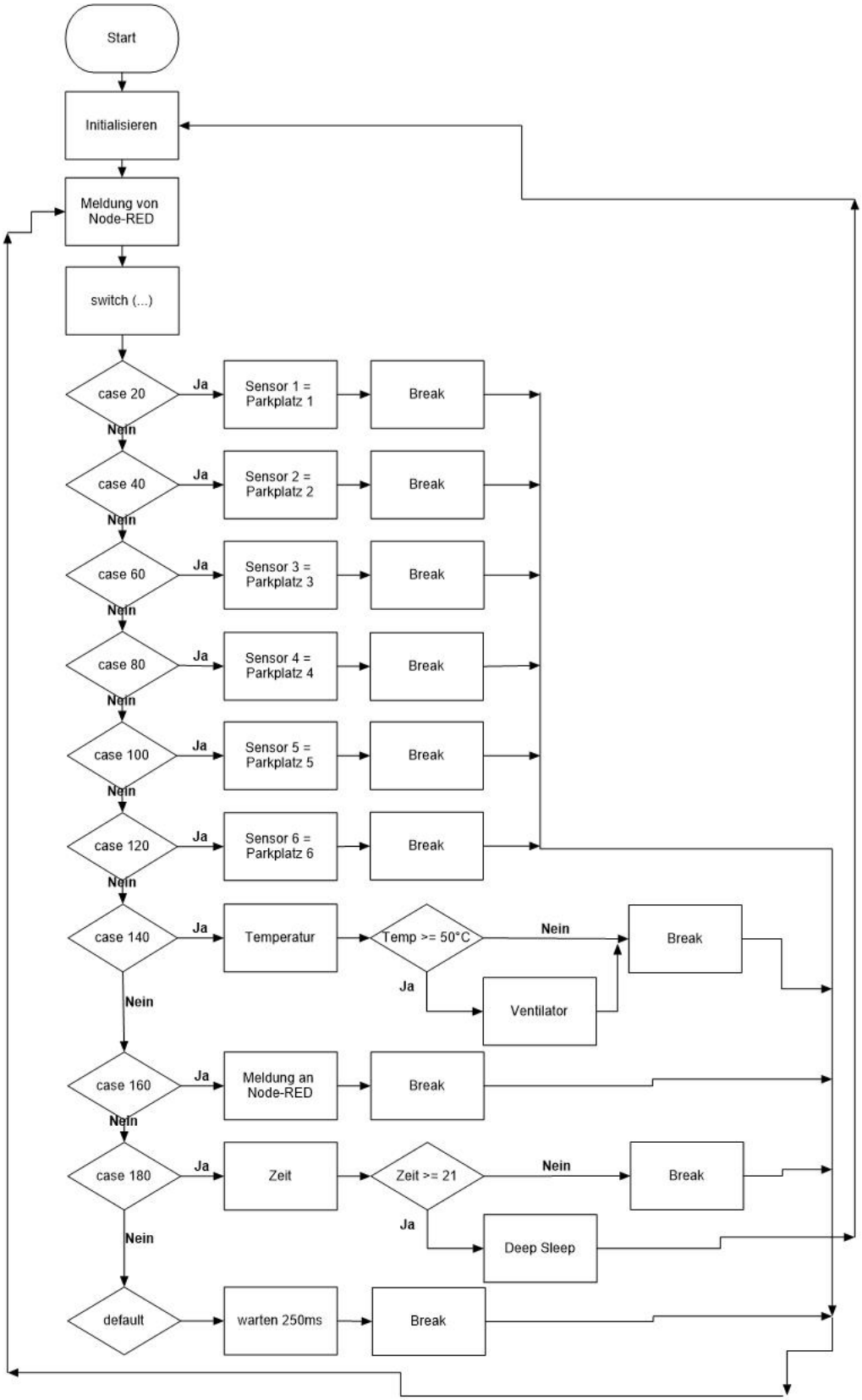


Abbildung 51: Flussdiagramm Software

Softwarecode des ESP32 und die einzelnen Funktionen

Nachfolgend werden einzelne Funktionen des Softwarecodes erläutert.
Es wurden zu den verwendeten Komponenten Bibliotheken eingebunden.
Nicht alle Punkte werden beschrieben, da das Grundgerüst für eine Kommunikation mit Node-RED meist gleich ist für jedes Projekt und dieses Grundkonstrukt auch im Internet nachgeschaut werden kann.
Der folgende Link ist ein gutes Beispiel hierfür:
https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide

Es wurde darauf geachtet, dass die allgemeinen Richtlinien für das Schreiben von Softwarecodes eingehalten wurden.
Genauer erläutert bedeutet das folgendes. Mit jeder Änderung am Softwarecode wird die Version hinauf gezählt und eine Kopie der alten Version gemacht. So kann jederzeit auf die ältere Version zurückgegriffen werden.
Der Softwarecode muss mit Kommentaren zum besseren Verständnis auskommentiert werden.
Die Deklaration der einzelnen Variablen findet für die globalen Variablen am Anfang statt und soll die Namen tragen, welche sie im Code vertreten.
Allgemein sollen die Namen der Variablen oder Methoden so gewählt werden, dass sie beschreiben, für was sie gebraucht werden.
Der Softwarecode soll strukturiert aufgebaut sein. Ähnliche Methoden sollen beieinander sein und nicht wild durcheinander.

Softwarecode Sensor Parkplatz

```
// Überprüfen ob ein Fahrzeug auf Parkplatz 1 steht
void sensor1()
{
  expander1.digitalWrite(P0, LOW);
  delayMicroseconds(2);
  expander1.digitalWrite(P0, HIGH);
  delayMicroseconds(20);
  expander1.digitalWrite(P0, LOW);
  distanceS1 = pulseIn(ECHO1, HIGH)/58;
  // Umwandlung des Wertes von Integer zu einem CharArray für die Übermittlung
  // ins Node-RED.
  char parking1 [10];
  itoa(distanceS1, parking1, 10);
  client.publish("esp32/sensor1", parking1);
  delay(500);
}
```

Abbildung 52: Softwarecode Sensor Parkplatz

Dieser Abschnitt des Codes dient dazu, dass der Ultraschallsensor die Datenerfassung ausführt und diese dann ins Node-RED übermittelt. Im Node-RED wird dann die übermittelte Distanz ausgewertet. Der obere Teil des Codes wurde aus dem Beispielcode für diesen Sensor übernommen.
Dieser Beispielcode kann auf der DFRobot Internetseite unter folgendem Link aufgerufen werden:
https://wiki.dfrobot.com/Weatherproof_Ultrasonic_Sensor_With_Separate_Probe_SKU_SEN0208
Dieser Codeabschnitt ist für alle sechs Ultraschallsensoren gleich und es werden nur die Pins sowie die Namen geändert.

Softwarecode Temperatur

```
// Überprüfen der Temperatur im Gehäuse
void casingTemperature()
{
  float temp = dht.readTemperature();
  // Umwandlung des Wertes von einem Float zu einem CharArray für die Übermittlung
  // ins Node-RED.
  char temperature [8];
  dtostrf(temp, 3, 2, temperature);
  client.publish("esp32/temp", temperature);
  delay(500);
  // Automatische Lüftersteuerung
  if(temp >= 50.0) // grösser 50°C sollen die Ventilatoren beginnen zu laufen um eine Überhitzung zu vermeiden
  {
    expander1.digitalWrite(P6, HIGH); // Hier werden beide Pins P6 und P7 gebraucht,
    expander1.digitalWrite(P7, HIGH); // um das Relais mit genügend Strom zu schalten
    delay(500);
  }
  else if((temp < 48.0) && (newFanState == "aus"))
  {
    expander1.digitalWrite(P6, LOW);
    expander1.digitalWrite(P7, LOW);
    client.publish("esp32/fan", "Ventilator aus");
    delay(500);
  }
}
```

Abbildung 53: Softwarecode Temperatur

Dieser Teil der Software soll dazu dienen, die Temperatur im Gehäuse wo der ESP32 eingebaut ist, zu überwachen. Durch diesen Code wird eine automatische Lüftersteuerung über die Temperatur gesteuert, sowie auch die aktuelle Temperatur ins Node-RED übermittelt und dort weiterverarbeitet. Wichtig hierbei ist zu wissen, dass zur Aktivierung des Relais zwei Pins des IO Expanders verwendet werden müssen, um genug Strom schalten zu können.

Softwarecode Zeit

```
else if (String(topic) == "esp32/time") // wenn die Unix Zeit gesendet wurde vom Topic esp32/time
{
 wakeupTime = strtoul(newTime.c_str(), NULL, 10); // Der Zeit String wird zu einem unsigned long geändert
 rtc.setTime(wakeupTime); // RTC Zeit wird gesetzt mit Wert des vom Node-RED gesendeten Wertes
 month = (rtc.getMonth()+1); // Abfrage des Monats +1 da die Zählung der Monate bei 0 beginnt
 if((month>=4)&&(month<=10)) // Abfrage des Monates zum Entscheiden ob Sommerzeit oder Winterzeit
 {
   summerTime = (wakeupTime + 7200); // GMT + 2 Stunden = Unsere Sommerzeit in der Schweiz
   rtc.setTime(summerTime);
 }
 else
 {
   winterTime = (wakeupTime + 3600); // GMT + 1 Stunde = Unsere Winterzeit in der Schweiz
   rtc.setTime(winterTime);
 }
}
```

Abbildung 54: Softwarecode Zeit

Bei diesem Teil wird eine Unix Zeit von einem Node-RED Timestamp Node gesendet. Da dies ein 10-stelliger Wert ist, muss dieser String in ein unsigned long geändert werden. Mit dieser Änderung auf unsigned long kann dieses System bis ca. in das Jahr 2078 ohne Probleme funktionieren. Danach wäre der Wert der Unix Zeit zu gross für das unsigned long.
Ausserdem wird in diesem Abschnitt unterschieden, ob die Unix Zeit von einem Monat mit Sommerzeit oder einem mit Winterzeit ist. Da die Tage des Wechsels auf Sommerzeit oder Winterzeit immer unterschiedlich ist, werden die Monate genommen. Darum ist in diesem Code die Sommerzeit von April bis und mit Oktober und die Winterzeit von November bis und mit März. Dem Entsprechend wird zur GMT-Zeit der Greenwich Mean Time entweder eine Stunde oder zwei Stunden dazu gezählt, damit es für unsere Zeitzone stimmt. Dadurch wird dann die aktuelle Tageszeit im ESP32 gesetzt. Diese Meldung wird im Node-RED jeden Tag um 07:15 Uhr abgesetzt.

```
case 180: counter = 0;
          actualTime = rtc.getTime(); // aktuelle Zeit des ESP32 auslesen
          actualTime.toCharArray(actualTimeESP, 10); // Zeit in ein Array speichern um an Node-RED zu senden
          client.publish("esp32/actualTime", actualTimeESP);
          if(rtc.getHour(true) >= 21) // Abfrage Uhrzeit 21:00 Uhr für Deep-Sleep
          {
            esp_sleep_enable_timer_wakeup(uS_TO_HOUR_FACTOR * 10); // 10 Stunden Deep-Sleep
            delay(100); // Delay das der ESP32 noch seine Arbeiten abschliessen kann bevor er schläft
            esp_deep_sleep_start();
          }
          break;
```

Abbildung 55: Softwarecode Deep-Sleep

Im case 180 wird die aktuelle Zeit ausgelesen des ESP32 und ins Node-RED übermittelt. Ausserdem wird, falls 21:00 Uhr ist, der Deep-Sleep aktiviert und der ESP32 schläft für 10 Stunden und resetet sich nach 10 Stunden beim Aufwachen.

Node-RED

Für die smarte Parkplatzüberwachung wird eine Oberfläche gebraucht, welche über ein Tablet oder einen PC erreichbar ist und die wichtigsten Informationen dem Sekretariat der TEKO Zürich anzeigt.
Die Node-RED Software ist ein grafisches Entwicklungswerkzeug, welches im Fall der smarten Parkplatzüberwachung auf dem Raspberry Pi installiert wurde.
Mit der IP-Adresse des Raspberry Pi kann die Node-RED Bearbeitungsoberfläche aufgerufen werden, bearbeitet und das User Interface aufgerufen werden.
Mit *http://IP-Adresse Raspberry Pi:1880* kann die Bearbeitungsoberfläche aufgerufen und bearbeitet werden.
Das User Interface wird sichtbar, wenn *http://IP-Adresse Raspberry Pi:1880/ui* eingegeben wird.
Für die Schrift der Oberfläche der TEKO Zürich, wurde die Farbe des TEKO Logos ermittelt und im Node-RED eingestellt. Um einen grösseren Kontrast zu haben, wurde der Hintergrund schwarz gemacht. Dies kennt man heutzutage auch als den Dark-Mode in allen gängigen Apps wie Instagram, Facebook oder auch bei verschiedenen Webbrowsern.

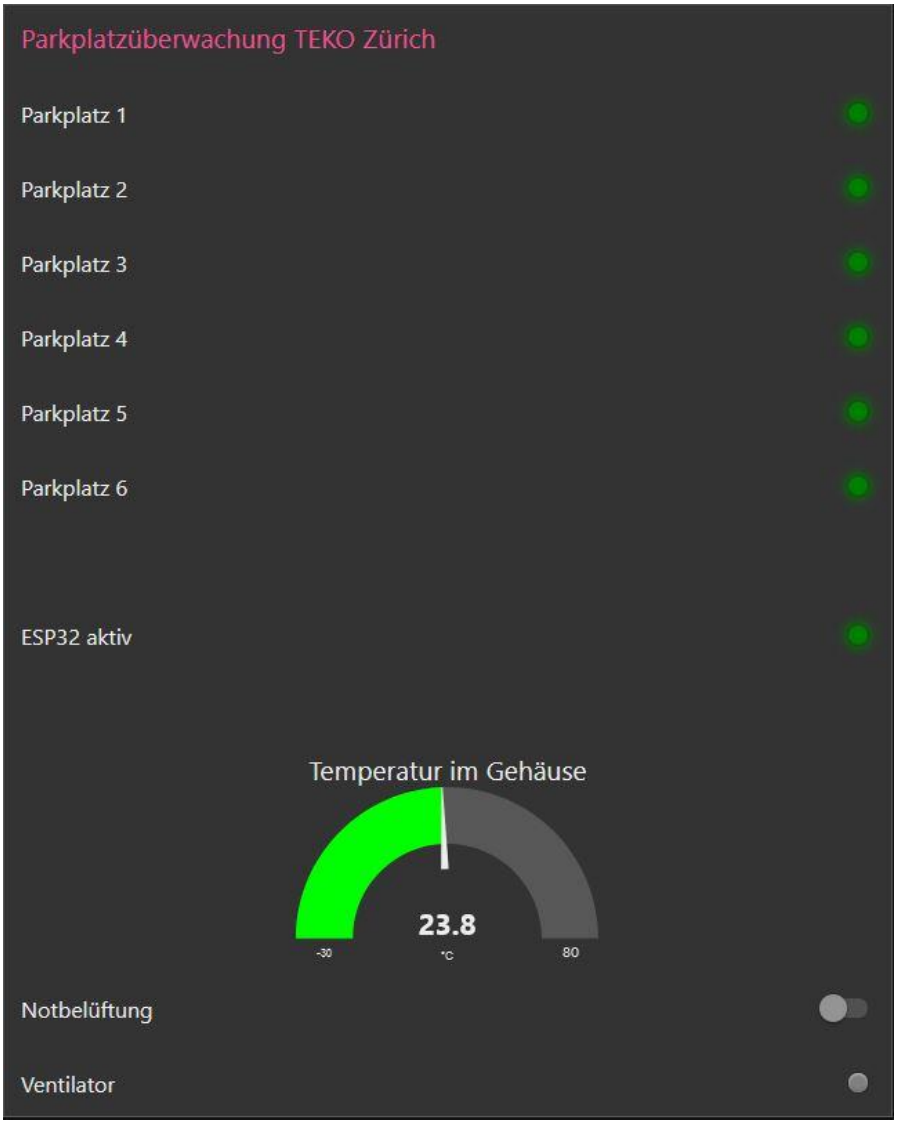


Abbildung 56: Benutzeroberfläche komplett

Node-RED und die einzelnen Funktionen

Die Benutzeroberfläche mit ihren einzelnen Funktionen, sowie die Prozesse im Hintergrund werden erläutert.
Für dieses Projekt musste die Bearbeitungsoberfläche um zwei Paletten erweitert werden.
Diese mussten unter Einstellungen/Palette/Installationen mit dem Namen „node-red-contrib-ui-led“ und ,,node-red-dashboard“ gesucht und installiert werden.
Die einzelnen Nodes, welche für dieses Projekt benutzt werden, wurden im 5. Semester im Fach Mikrokontroller erläutert. Die Funktionen der einzelnen Nodes werden in dieser Diplomarbeit nicht erklärt.
Für den Testaufbau wird in der Bearbeitungsoberfläche immer wieder das Debug-Node zu erkennen sein. Dieses Node ist nur für Testzwecke und wird in der Version für die TEKO nicht mehr vorhanden sein.
Für das Sekretariat steht eine Anleitung separat zur Verfügung, in welcher erklärt ist, was die einzelnen Funktionen sind und was in einem Fehlerfall gemacht werden kann.

Die Endversion der Node-RED Oberfläche wird auf einem USB-Stick verfügbar sein für Erweiterungen durch die TEKO Zürich.

Node-RED Parkplatz 1 bis 6

Die Parkplätze 1 bis 6 mit ihren LED's zeigen an ob ein Parkplatz frei oder besetzt ist.
Die Werte, welche die Ultraschallsensoren an den ESP32 übergeben, sendet dieser über WLAN mit dem entsprechenden Topic an den MQTT Broker (Raspberry Pi). Der Broker übergibt diese Daten dann an die Node-RED Oberfläche. Dort ist eine Funktion für jeden Parkplatz hinterlegt, welche entscheidet, ob ein Parkplatz frei oder besetzt ist.
Da die Funktion für alle Parkplätze gleich ist, wird nur Parkplatz 1 erläutert.

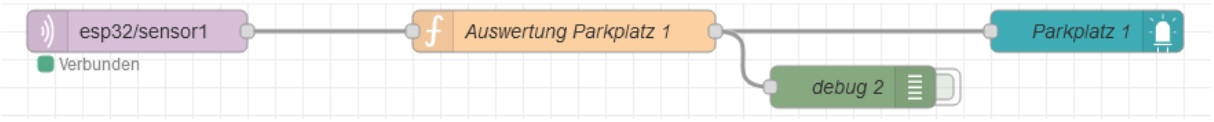


Abbildung 57: Funktion Parkplatz 1 mit debug

Für den Testaufbau wird die obere Abbildung verwendet.
Für den Parkplatz 1 sendet der ESP32 die gemessene Distanz am Sensor 1 an den MQTT-Broker mit dem Topic esp32/sensor1, diese Distanz kommt als Nachricht im Funktions-Node Auswertung Parkplatz 1 an. Dort wird über eine if/else Abfrage entschieden, ob der Parkplatz 1 frei oder besetzt ist und sendet an den LED-Node mit Namen Parkplatz 1 ein „true" für Parkplatz frei oder ein „false", wenn der Parkplatz besetzt ist. Im Testaufbau wird angenommen, dass der Parkplatz ab einer Distanz von 50cm vom Sensor entfernt als frei anzusehen ist.

```
1  if(msg.payload >=50)
2  {
3      msg.payload = true;
4  }
5  else
6  {
7      msg.payload = false;
8  }
9  return msg;
```

Abbildung 58: Abfrage Parkplatz frei?

Die Parkplatz LED's auf der Benutzeroberfläche ändern ihre Farben auf Grün für frei und auf Rot für besetzt.

Abbildung 59: Parkplätze frei

Abbildung 60: Parkplätze besetzt

Node-RED ESP32 aktiv

Die Funktion ESP32 aktiv wurde in die Benutzeroberfläche eingebunden, um dem Sekretariat anzuzeigen, falls der ESP32 nicht mehr senden kann. Diese Funktion ist wichtig, weil wenn der ESP32 keine Daten mehr sendet die Benutzeroberfläche die letzten erhaltenen Daten anzeigt. Diese müssen nicht der Realität entsprechen. Es kann sein, dass der ESP32 defekt ist oder eine der Speisungen nicht mehr vorhanden ist.

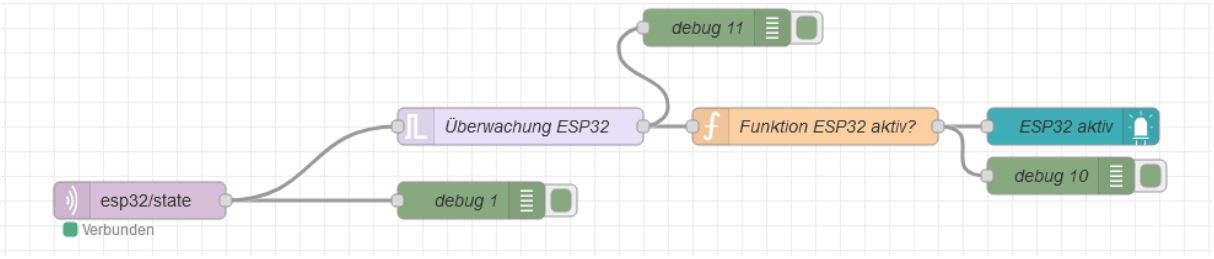


Abbildung 61: Funktion ESP32 aktiv mit debug

Für den Testaufbau wird die obere Abbildung verwendet.
Der ESP32 sendet rund alle 45 Sekunden auf das Topic esp32/state die Nachricht „ESP32 aktiv“. Sollte diese Nachricht während zwei Minuten nicht erneut gesendet werden, gibt die Trigger Funktion mit dem Namen Überwachung ESP32 ein „false“ als Nachricht weiter an das Funktions-Node Funktion ESP32 aktiv. Eine if/else Abfrage wertet aus, ob der ESP32 sich gemeldet hat oder nicht. Es wird ein „true“ für ESP32 aktiv oder ein „false“ für nicht aktiv an das LED-Node mit dem Namen ESP32 aktiv gesendet.

```
1   if (msg.payload == "ESP32 aktiv")
2   {
3      msg.payload = true;
4   }
5   else
6   if(msg.payload = false)
7   {
8       msg.payload = false;
9   }
10  return msg;
```

Abbildung 62: Abfrage ESP32 aktiv?

Die ESP32 aktiv LED auf der Benutzeroberfläche kann ihre Farbe auf Grün für aktiv und auf Rot für inaktiv ändern.

Abbildung 63: ESP32 aktiv

Abbildung 64: ESP32 inaktiv

Node-RED Temperatur und Ventilatoren

Als weitere Sicherheit wird über den Temperatursensor die aktuelle Temperatur im Gehäuse, in welchem sich der ESP32, sowie die Akkus und die Zusatzhardware befinden, gemessen.
Im Softwarecode des ESP32 ist eine automatische Ventilatorsteuerung einprogrammiert. Sollte diese Automatik nicht funktionieren und die Temperatur sinkt nicht, sowie die Ventilatoren laufen nicht an, obwohl der Grenzwert erreicht ist, kann über das Tablet der Schalter auf der Benutzeroberfläche mit dem Namen Notbelüftung die Ventilatoren im Gehäuse manuell eingeschaltet werden.

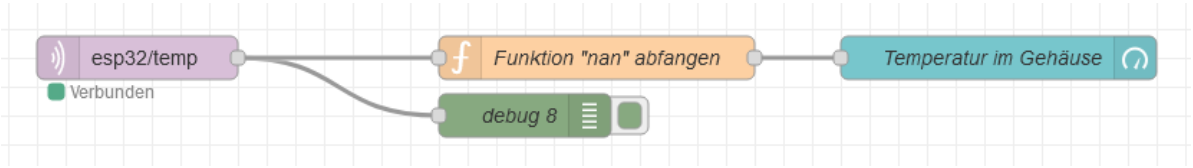


Abbildung 65: Temperatur

Der Temperatursensor sendet via ESP32 und MQTT Broker auf das Topic esp32/temp die aktuelle Temperatur im Gehäuse. Diese wird auf der Benutzeroberfläche anhand einer Anzeige angezeigt. Die Anzeige kann ihre Farbe in den Farben Blau, Grün und Rot ändern. Blau für Temperaturen zwischen -30°C bis 0°C. Grün für Temperaturen zwischen 0°C und bis 40°C und Rot für Temperaturen über 40°C.
Da der Temperatursensor immer mal wieder die Nachricht „nan“ sendet, wurde die Funktion „nan“ abfragen eingebaut. Diese Funktion soll im Falle, dass der Sensor keine brauchbaren Daten liefert, den letzten Wert als Nachricht an das Gauge-Node weitergeben. Dadurch wird gewährleistet, dass immer einen Wert angezeigt wird, auch wenn „nan“ gesendet wird.

```
1   if(msg.payload == "nan")
2   {
3       msg = lastmsg;
4   }
5   else
6   {
7      var lastmsg = msg.payload;
8   }
9   return msg;
```

Abbildung 66: Abfrage "nan"

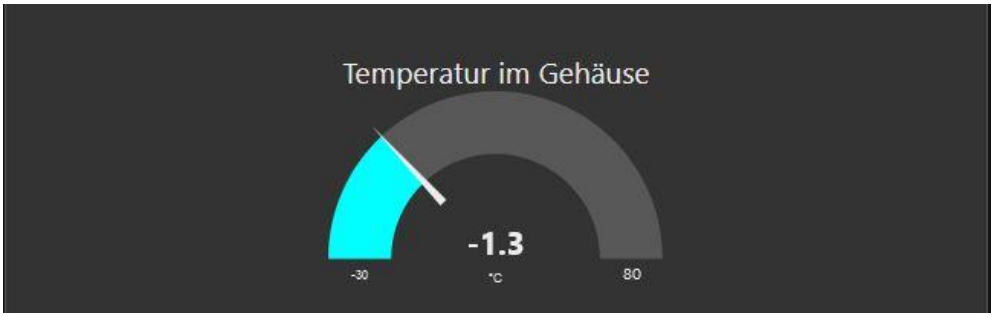


Abbildung 67: Temperatur -30 bis 0°C

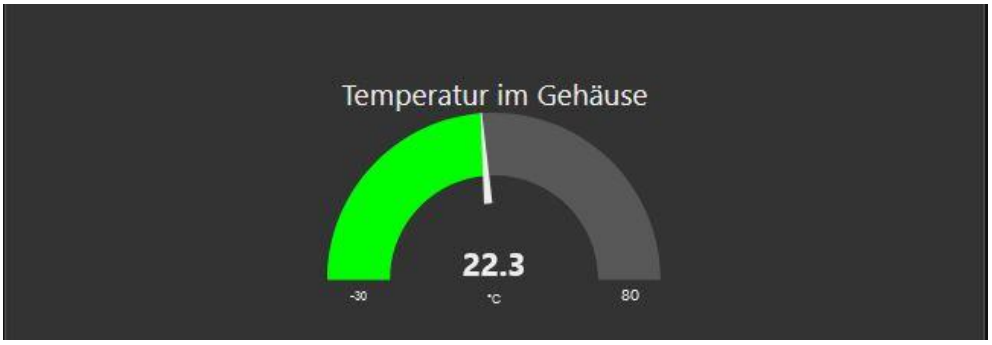


Abbildung 68: Temperatur 0 bis 40°C

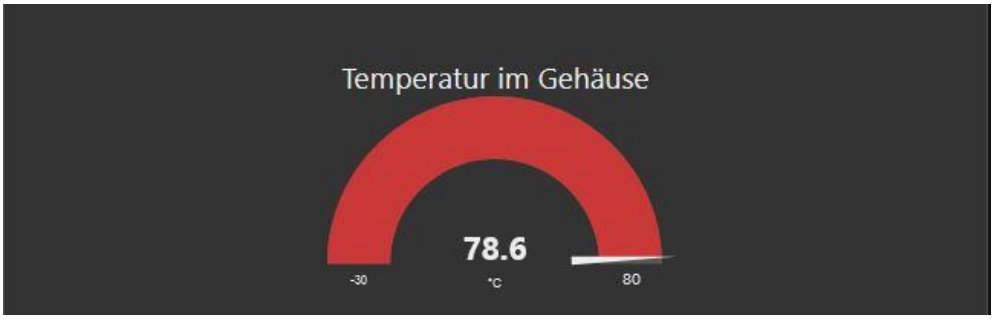


Abbildung 69: Temperatur 40 bis 80°C

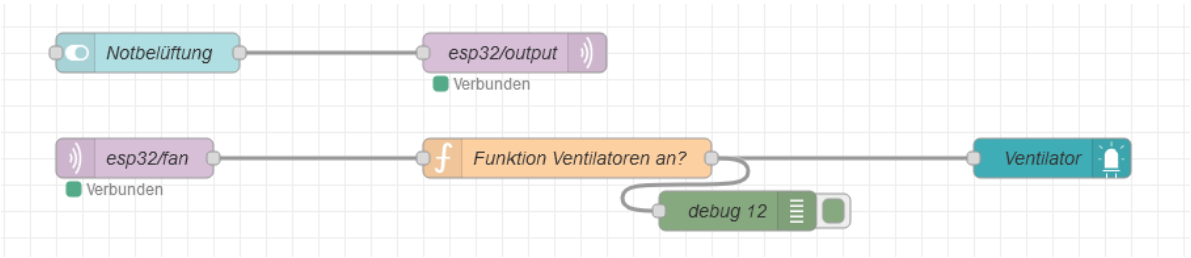


Abbildung 70: Notbelüftung und Ventilatoren

Die Abbildung Notbelüftung und Ventilatoren wird für den Testaufbau verwendet. Der Switch-Node Notbelüftung sendet die Befehle „ein“ und „aus“. Der ESP32 hat den Topic esp32/output abonniert und bekommt dadurch den Befehl die Ventilatoren ein- oder auszuschalten. Schaltet dieser die Ventilatoren ein oder aus, wird eine Nachricht zurück ans Topic esp32/fan gesendet. Die Nachricht wird im Funktions-Node „Namen Funktion Ventilatoren an? “ mit einer if/else Abfrage überprüft. Wenn die Nachricht „Ventilator an“ detektiert wird, sendet die Funktion ein „true“ an den LED-Node mit dem Namen „Ventilator“. Ansonsten wird „false“ gesendet.

```
if (msg.payload == "Ventilator an")
{
    msg.payload = true;
}
else
{
    msg.payload = false;
}
return msg;
```

Abbildung 71: Abfrage Ventilator an?

Der Schalter auf der Benutzeroberfläche ändert seine Farbe, wenn er eingeschaltet wird in die Farbe des TEKO-Schriftzugs. Die Ventilator LED kann ihre Farbe auf Blau für Ventilatoren aktiv und auf Rot für inaktiv ändern.

Abbildung 72: Notbelüftung ein, Ventilator aktiv

Abbildung 73: Notbelüftung aus, Ventilator inaktiv

Node-RED Zeit

Um den ESP32 zu einer bestimmten Zeit in den Deep-Sleep versetzen zu können, kann über Node-RED ein Zeitstempel gesendet werden an den ESP32, welcher dann auf dem ESP32 verarbeitet wird.

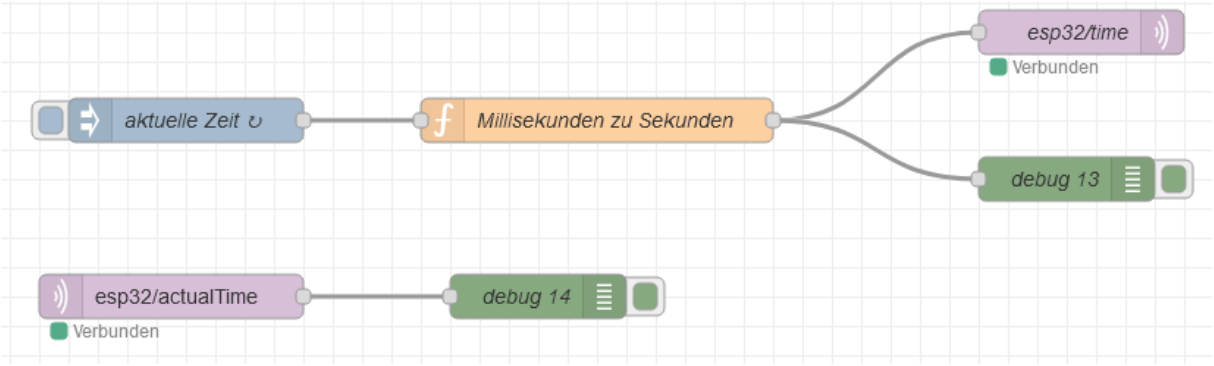


Abbildung 74: Zeit

Die Abbildung oben zeigt, wie die Zeit an den ESP32 gesendet wird und was der ESP32 zurücksendet. Mit dem Inject-Node kann ein Zeitstempel zu jeder Tageszeit gesendet werden. In diesem Projekt wurde dieses Node so eingestellt, dass es jeden Morgen um 07:15, nachdem der ESP32 aufgewacht ist, die aktuelle Zeit sendet. Dieses Node sendet einen Unix Zeitstempel in Millisekunden. Zu beachten ist, dass dieser Unix Zeitstempel die GMT-Zeit (Greenwich Mean Time) hat und diese nicht unserer Zeitzone entspricht. Die Unixzeit zählt die Zeit seit dem 1. Januar 1970 um 00:00 Uhr. Unsere Zeitzone hier in der Schweiz ist im Sommer zwei Stunden mehr wie die GMT-Zeit und im Winter eine Stunde. Dieser Punkt wird im Softwarecode des ESP32 abgefangen und entsprechend hinzugezählt, je nach dem welcher Monat ausgelesen wird.
Die Zahl, welche an den ESP32 gesendet wird, ist so gross, dass diese Unix Zeit zuerst in Node-RED in Sekunden gewandelt wird mit dem Funktions-Node „Millisekunden zu Sekunden“.

```
1  msg.payload = parseInt((msg.payload / 1000));
2  return msg;
```

Abbildung 75: Millisekunden zu Sekunden

Sobald der ESP32 seine RTC-Zeit angepasst hat, sendet dieser auf das Topic esp32/actualTime die aktuelle Zeit des ESP32. Diese Funktion ist zur Kontrolle, dass auch die richtige Zeit gesendet wird. Hat jedoch keinen Einfluss auf die Benutzeroberfläche, sondern ist nur für eine Allfällige Prüfung, wenn etwas sein sollte.

Der Prototyp

Bevor der richtige Aufbau in der TEKO Zürich aufgebaut wird, wird ein Prototyp erstellt.
Dieser soll dazu dienen, um allfällige Probleme frühzeitig erkennen zu können.
Auch hilft dieser Prototyp den Ablauf richtig zu organisieren.
Es werden mehrere Gehäuse verwendet, in welchen die Sensoren, die Sensorboards sowie die Elektronik verbaut werden.
Die Kabellängen werden jenen des realen Aufbaus in der TEKO nachempfunden.
So kann verhindert werden, dass die Kabellängen Einfluss auf die Sensoren nehmen.
Lange Leitungen können Störungen hervorrufen durch ihren Leitungswiderstand. Durch diesen Prototypen soll verhindert werden, dass der Querschnitt für das gewählte Kabel zu tief ist.
Da bei Kupferleitungen die Regel gilt, dass umso grösser der Querschnitt, desto länger darf das Kabel sein.
Zu beachten ist, dass alle Materialen die direkten Kontakt zu den Witterungen haben so gewählt werden, dass diese den Witterungen in Glattbrugg stand halten.

Abbildung 76: Prototyp

In den nächsten Abschnitten wird erläutert, wie der Prototyp sich zusammensetzt.
Dieser ist unterteilt in Sensorbox, Sensorboardbox, Elektronikbox und Solarpanel.
Dieser komplette Aufbau wurde ohne fremde Hilfe erschaffen.

Elektroschema des Prototyps

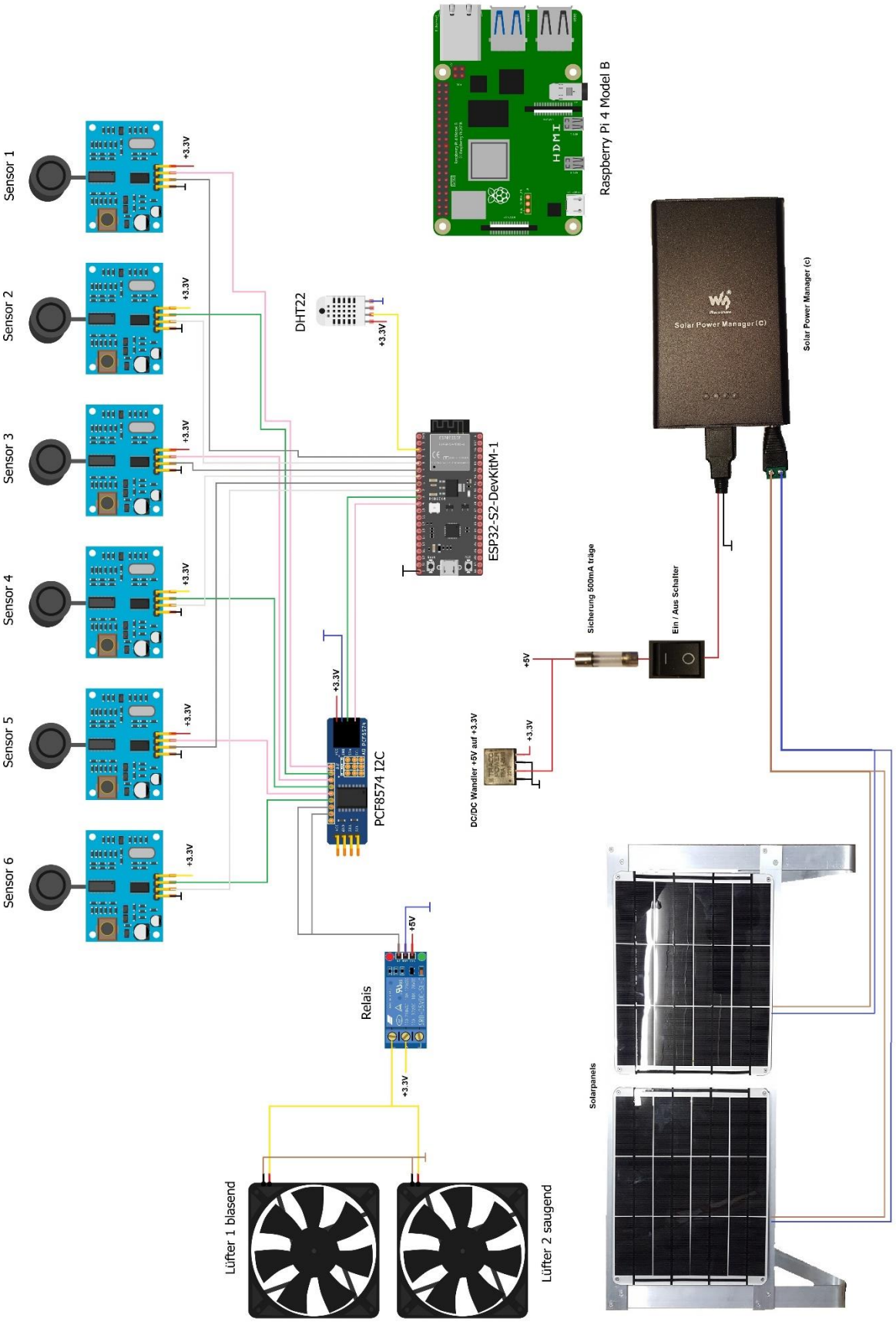


Abbildung 77: Elektroschema des Prototyps mit Originalkabelfarben

Sensorbox

Das einfachste Gehäuse in der Herstellung ist die Sensorbox. Von diesen Sensorboxen gibt es sechs an der Zahl. In diesem Gehäuse befindet sich der Ultraschallsensor. Die nächsten drei Bilder zeigen den Weg zur fertigen Sensorbox.

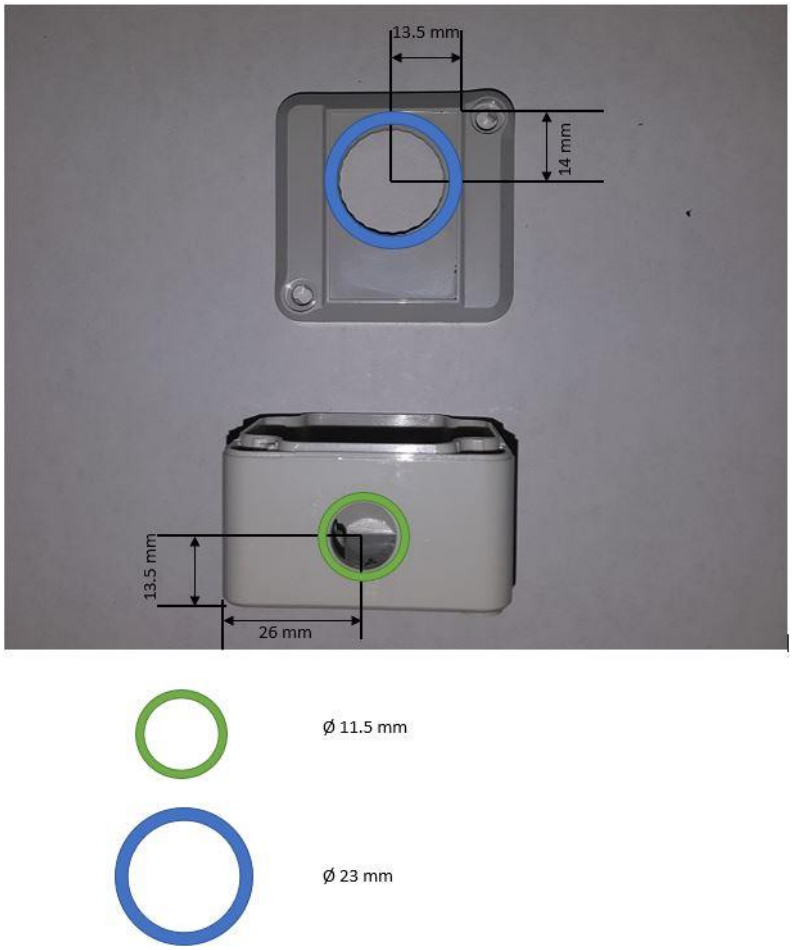


Abbildung 78: Sensorbox vermasst

Die Stopfen im unteren Bild müssen bearbeitet werden.
Weil der Stecker des Sensors ein so grosses Loch braucht um diesen durch zu fädeln, wird dieser Stopfen als Abschluss eingebaut.
Dieser Stopfen wird eingekauft wie bei Punkt 1. Danach wird in Punkt 2 ein Loch mit Durchmesser 3mm rein gebohrt. In Punkt 3 wird dieser bis auf die letzte Lippe abgeschnitten und bei Punkt 4 noch auf einer Seite durchgeschnitten um das Kabel einzufädeln.

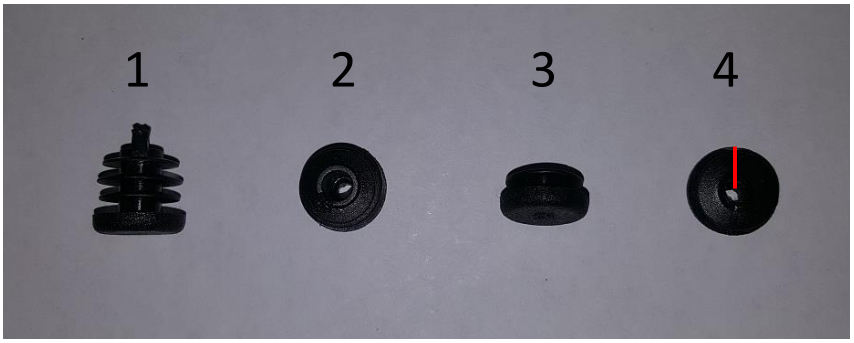


Abbildung 79: Stopfen bearbeitet

Nach erfolgreichem Zusammenbau ist die Sensorbox fertig.

Abbildung 80: Senorbox

Sensorboardbox

Als nächstes folgt die Sensorboardbox. Es gibt drei dieser Boxen und in jedem Gehäuse befinden sich zwei Sensorboards. Die folgenden Bilder zeigen den Weg bis zur fertigen Box.

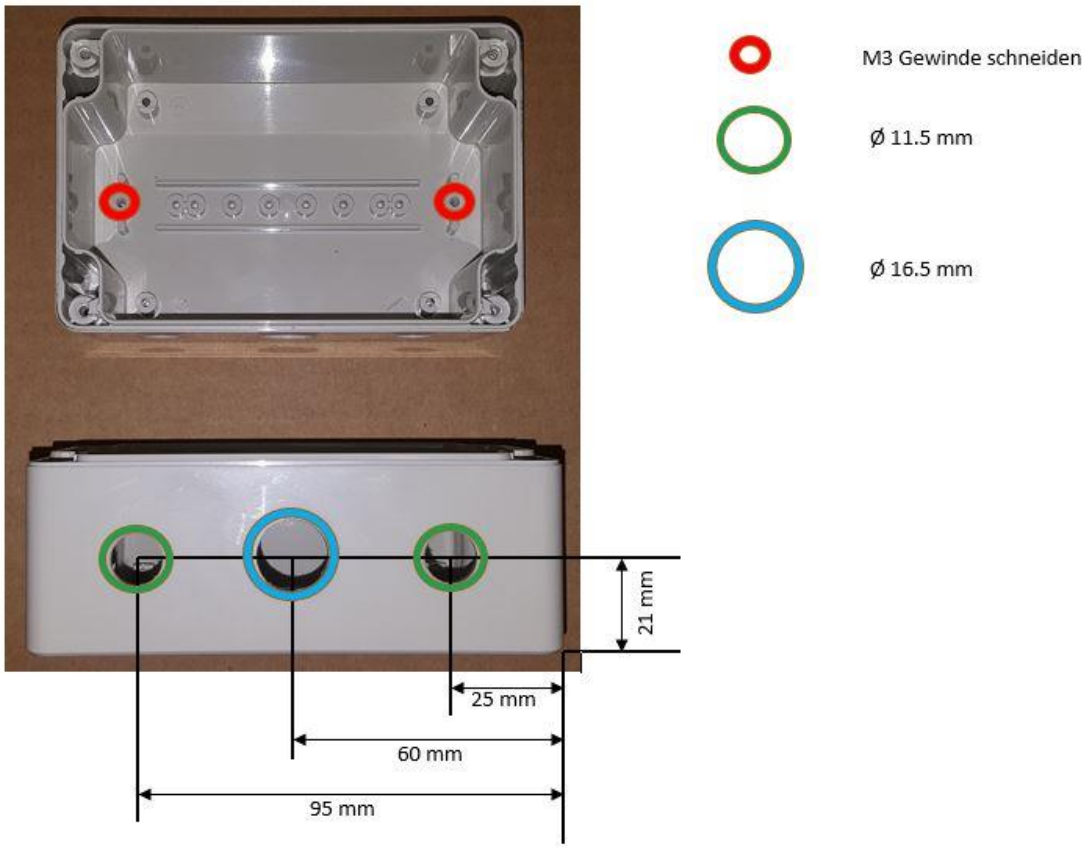


Abbildung 81: Senorboardbox vermasst

Wie bei der Sensorbox braucht es auch hier bei den 11.5mm Löchern die Stopfen. Sechs Stück an der Zahl für die drei Gehäuse. Da die Sensorboards mit dem Stecker zu hoch für das Gehäuse sind, wurde ein Winkel konstruiert um die Sensorboards statt liegend im Gehäuse zu montieren stehend zu verbauen. Dieser muss am Schluss noch modifiziert werden, weil eines der beiden Sensorboards verdreht bestückt wird, damit die Anschlüsse mittig sind für das acht Polige Kabel, welches die Speisung und die Signale Echo und Trigger führt.

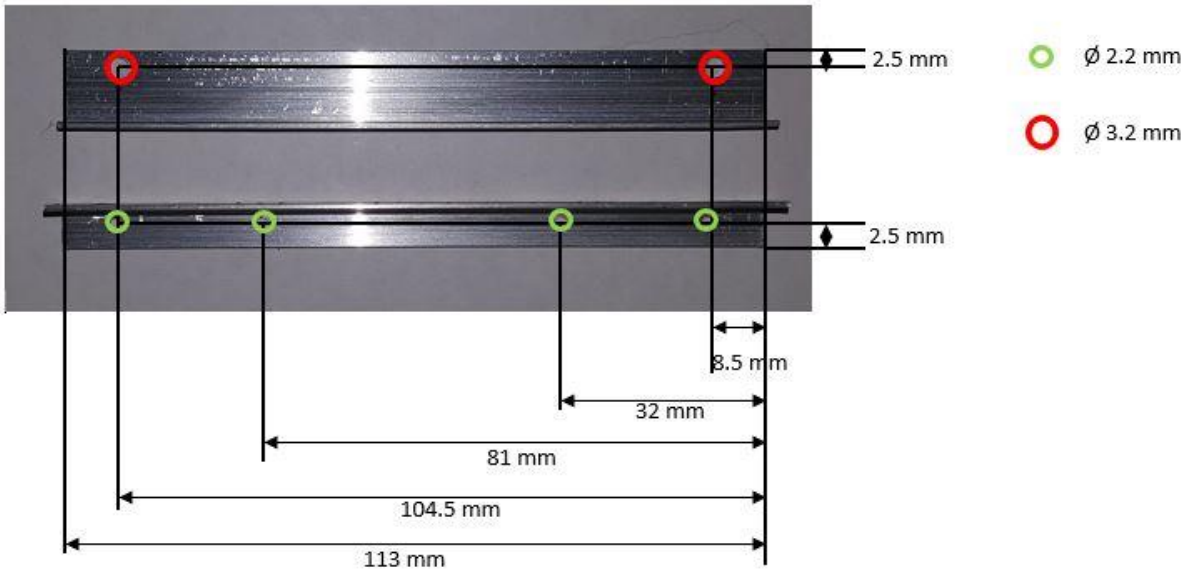


Abbildung 82: Winkel bemasst

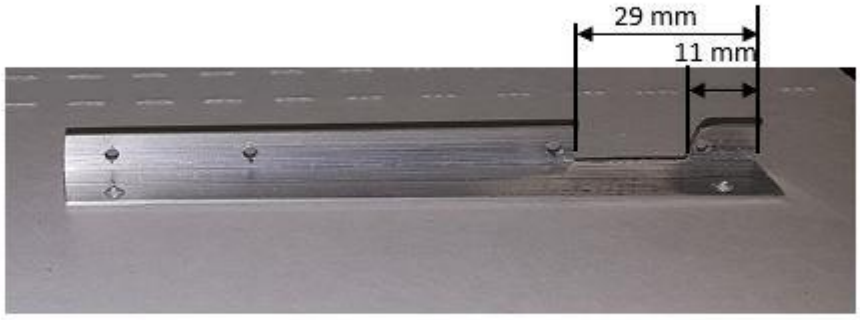


Abbildung 83: Winkel modifiziert

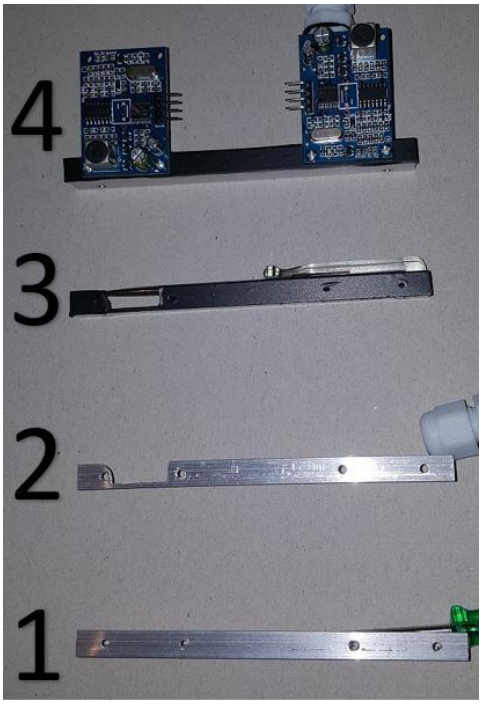


Abbildung 84: Weg zum fertigen Winkel

In diesem Bild ist in vier Schritten gezeigt, wie der Winkel entsteht. Dieser wird bevor die Boards montiert werden, an der Kante, wo die Boards angeschraubt werden, mit Isolierband überzogen. Dies erfolgt in Punkt 3. Punkt 4 ist der einbaufertige Winkel mit den beiden Senorboards dran.

Somit können die beiden Sensoren an die Sensorboards angeschlossen werden. Die Kabelfarben für die Sensorboards wurden wie folgt gewählt.

Ungerade Parkplatznummern haben folgende Leitungsfarben:

- + = rot
- Trig = pink
- Echo = grau
- GND = blau

Gerade Parkplatznummern haben folgende Leitungsfarben:

- + = gelb
- Trig = grün
- Echo = weiss
- GND = braun

Nach erfolgreichem Zusammenbau sieht die Box wie im unteren Bild aus.

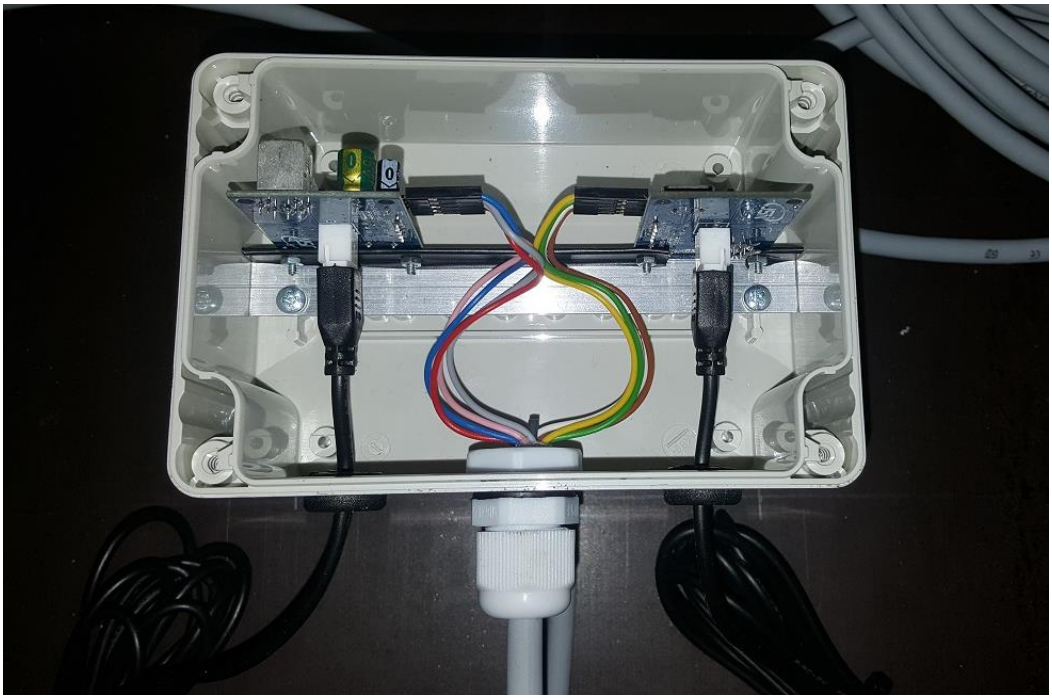


Abbildung 85: Sensorboardbox

Elektronikbox

Die Elektronikbox ist die komplexeste aller drei Boxen. Dort drin befinden sich der Print mit den Speisungen und den Sensoren und deren Anschlüsse sowie der Mikrokontroller. Es befinden sich ausserdem der Solar Manager sowie die Ventilatoren darin. In den nächsten Bildern ist die Entstehung dieser Elektronikbox erklärt.

Als erstes muss das Gehäuse mechanisch bearbeitet werden, um die Ventilatoren betreiben zu können, wie auch die ganzen Kabel und der Ein / Aus Schalter. Es müssen in den Boden ausserdem M4 Gewinde geschnitten werden, um die Leiterplatte und die Platte, welche den Solar Manager hält, befestigen zu können.

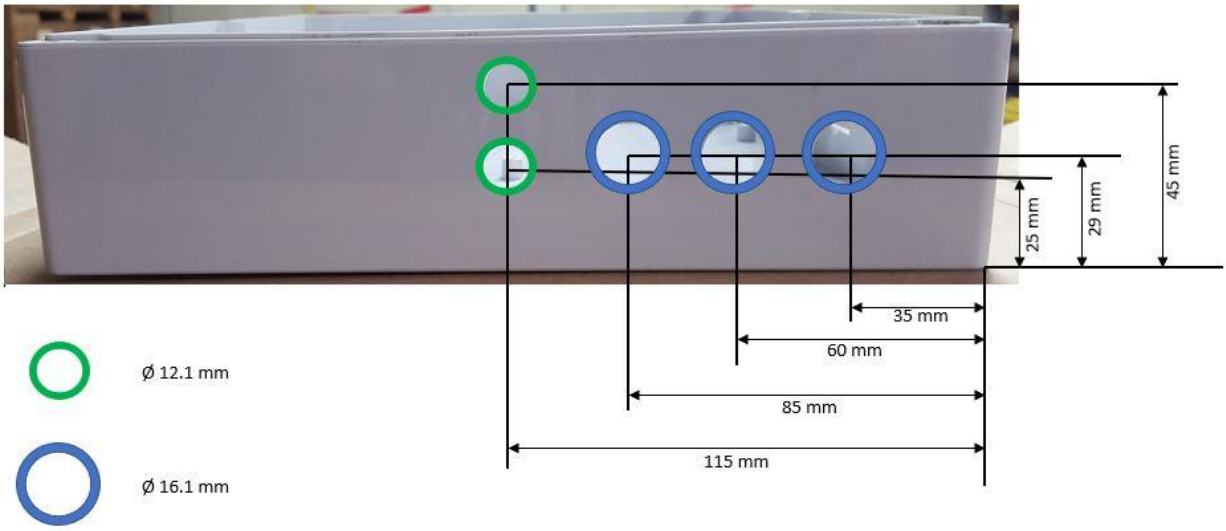


Abbildung 86: Elektronikbox untere Seite vermasst

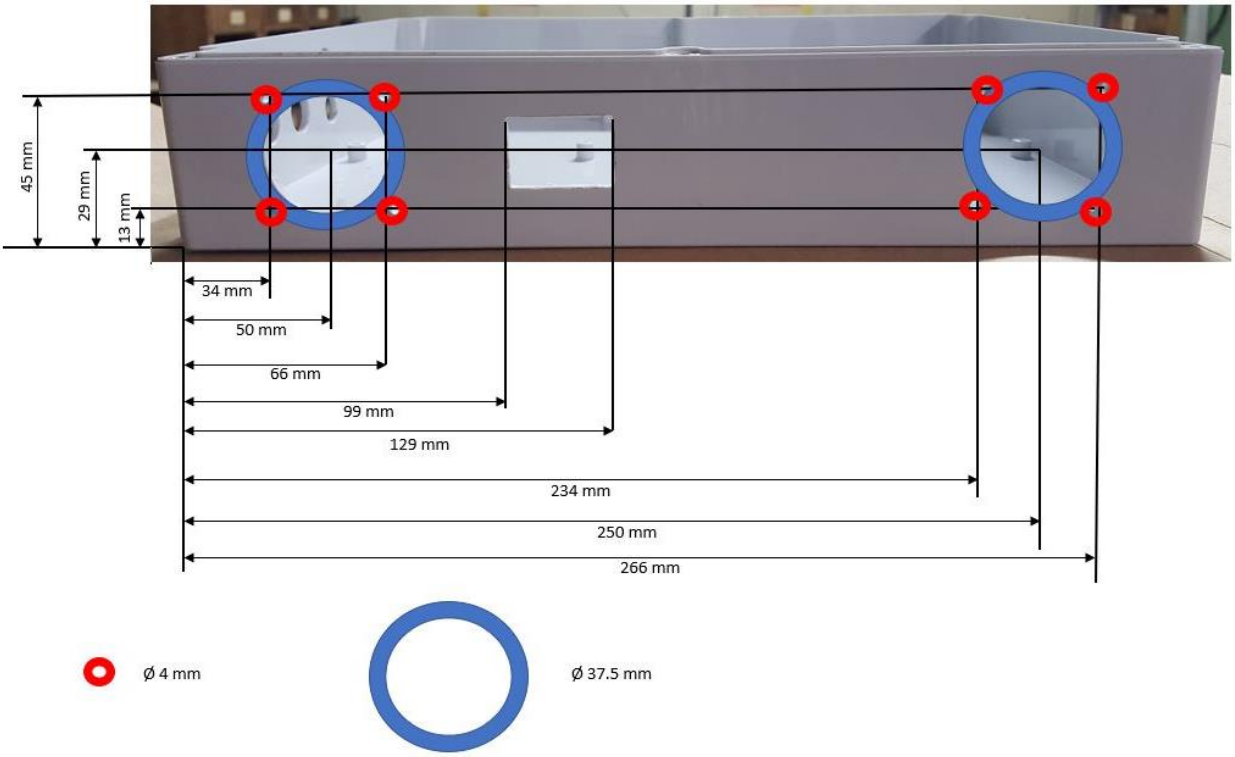


Abbildung 87: Elektronikbox rechte Seite vermasst

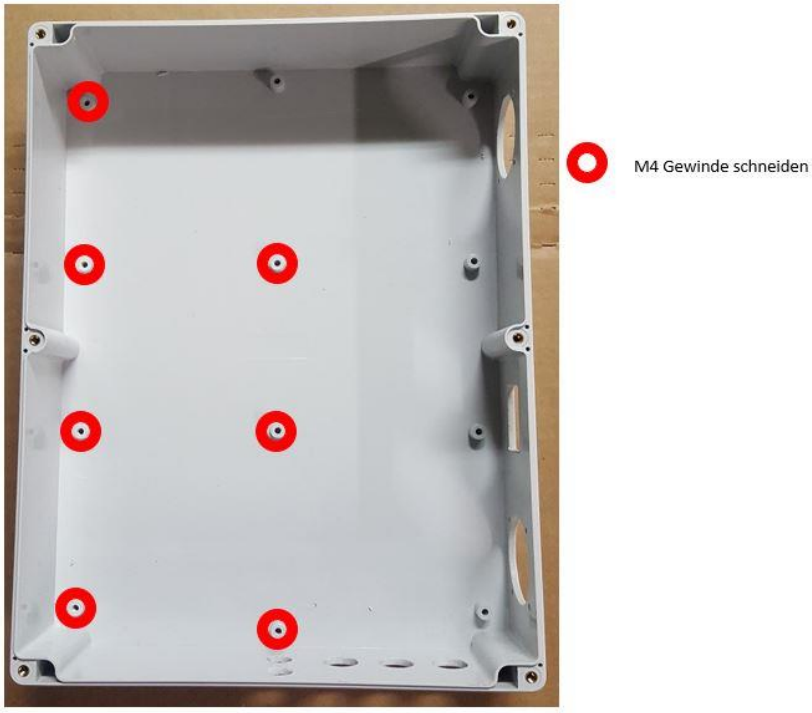


Abbildung 88: M4 Gewinde im Boden der Elektronikbox

Um den Solar Manager zu befestigen braucht es eine Platte. Hierzu wurde eine Aluplatte bearbeitet und mit Klettbändern versehen.

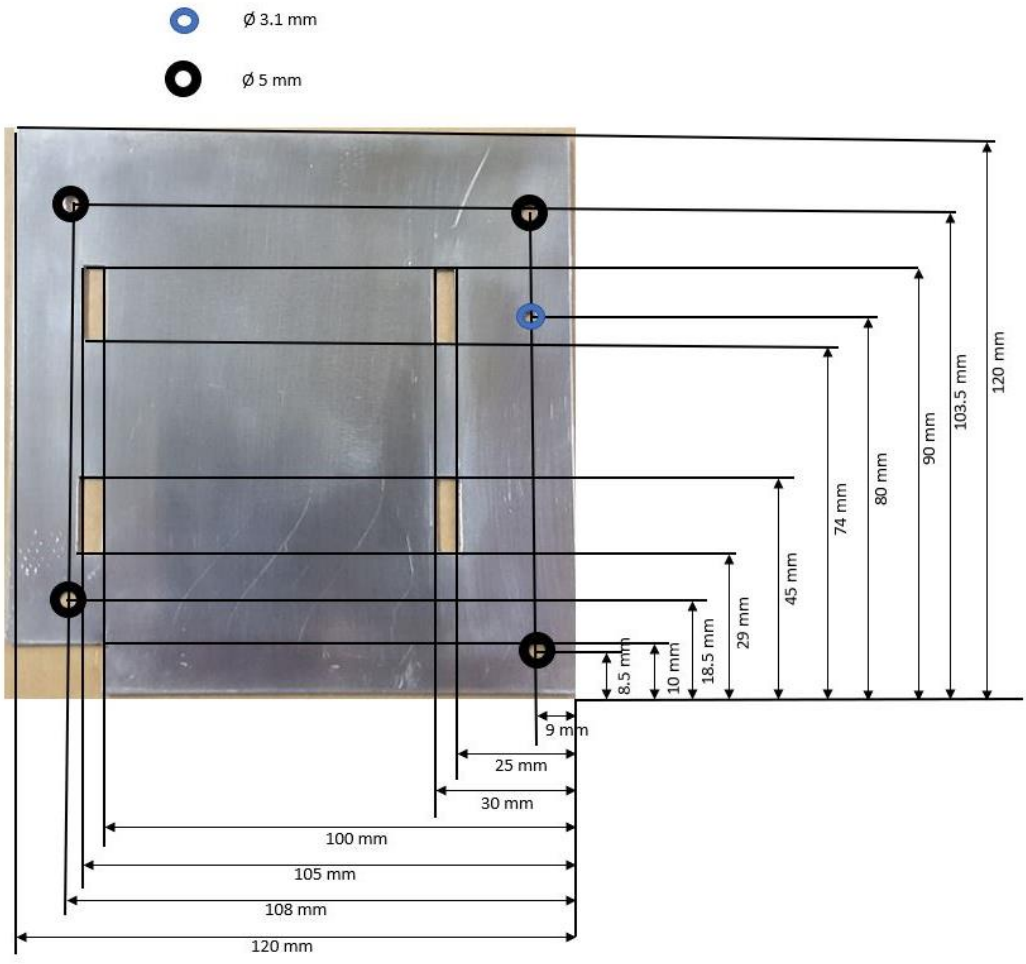


Abbildung 89: Platte Solar Manager vermasst

Abbildung 90: Befestigung Solar Manager

Die Leiterplatte muss, bevor diese bestückt werden kann, zuerst mechanisch bearbeitet werden.

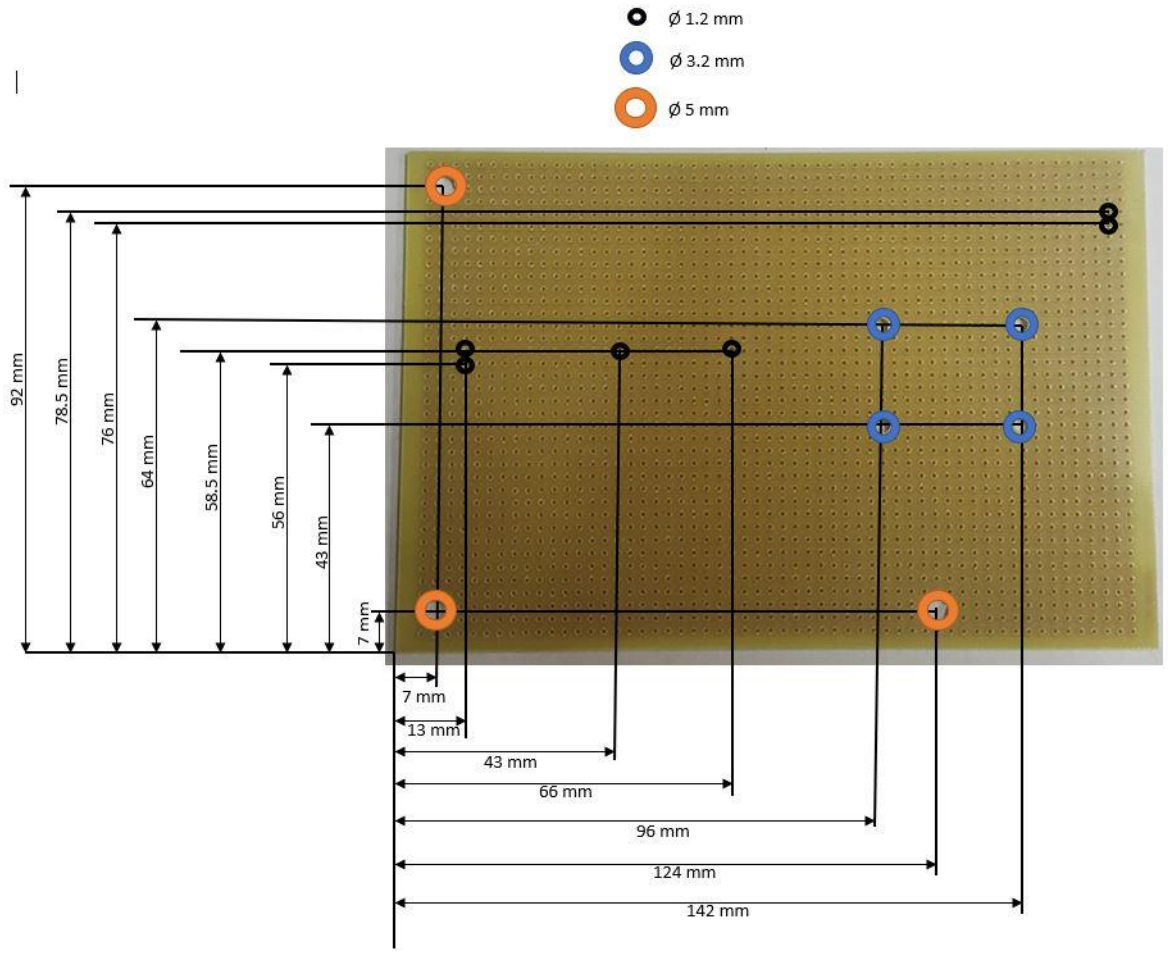


Abbildung 91: Leiterplatte bemasst

Nachdem die Leiterplatte die benötigten Löcher hat, kann diese bestückt werden.
Die Leiterplatte ist so bestückt, dass jeder Zeit die Bauteile ausgewechselt werden können, wenn ein Bauteil kaputt gehen würde oder ersetzt werden soll. Es ist sehr wichtig, dass dieses System nachhaltig gebaut wird. Es soll bei einem Problem nicht der ganze Print erneuert werden müssen. Darum hat es auch genügend freie Pins, um weitere Sensoren und andere Bauteile an diese Leiterplatte anzuhängen. Der Grossteil der Verbindungen auf dem Print sind mit Jumperkabel gemacht. Lediglich die Speisung ist auf dem Print fest verdrahtet. Auch sind die wichtigsten Anschlüsse beschriftet. Es wurde darauf geachtet, dass die Kabelfarben gleich den Farben im kompletten Schema sind.

Es folgt ein Bild der komplett bestückten Leiterplatte. In diesem Zustand ist die Leiterplatte einbaufertig ins Gehäuse und kann weiter verkabelt werden, mit den Sensorboards sowie mit dem Solar Manager und den Ventilatoren.

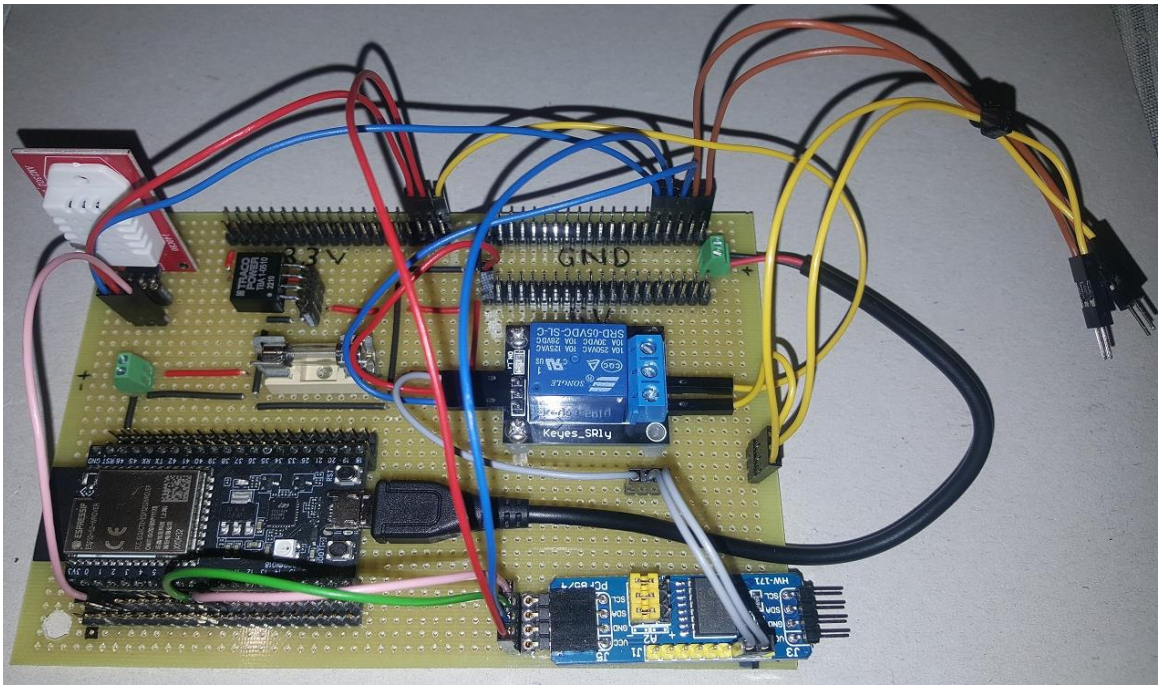


Abbildung 92: Leiterplatte komplett bestückt

Nachdem alle Teile bereit sind, können die ganzen Komponenten der Elektronikbox verkabelt werden.
Nach erfolgreichem Zusammenbau ist die Elektronikbox fertig und sieht wie in dem untenstehenden Bild aus.

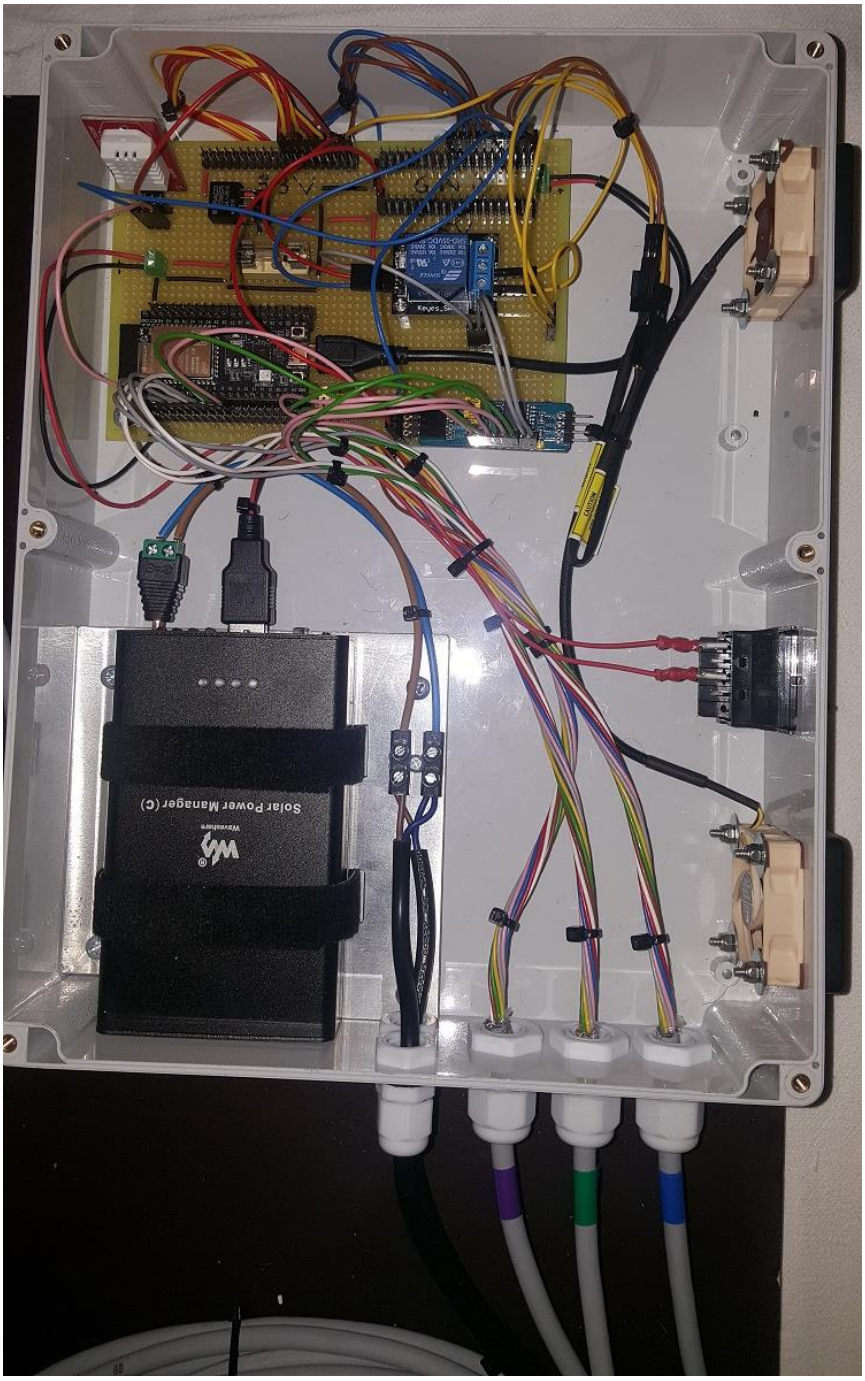


Abbildung 93: Elektronikbox

Solarpanel

Die beiden Solarpanel sollen bei der TEKO erhöht auf einem Gestell verbaut werden, damit diese volle Sonneneinstrahlung erhalten.
Dafür wurde ein Rahmen gebaut, welcher so gebogen und zusammengebaut ist, dass beide Solarpanel genau gleich ausgerichtet sind. Dies muss zwingend beim Parallelbetrieb gewährleistet sein.

Abbildung 94: Halterung mit Solarpanels

Wichtig ist für die verbauten Solarpanels, dass der Solar Manager auf 6V eingestellt wird.
Dazu muss die Leiterplatte des Solar Managers ausgebaut werden. Dort gibt es auf der unteren Seite der Leiterplatte einen Dip-Switch. Dieser muss in diesem Fall bei 6V auf ON gestellt werden.

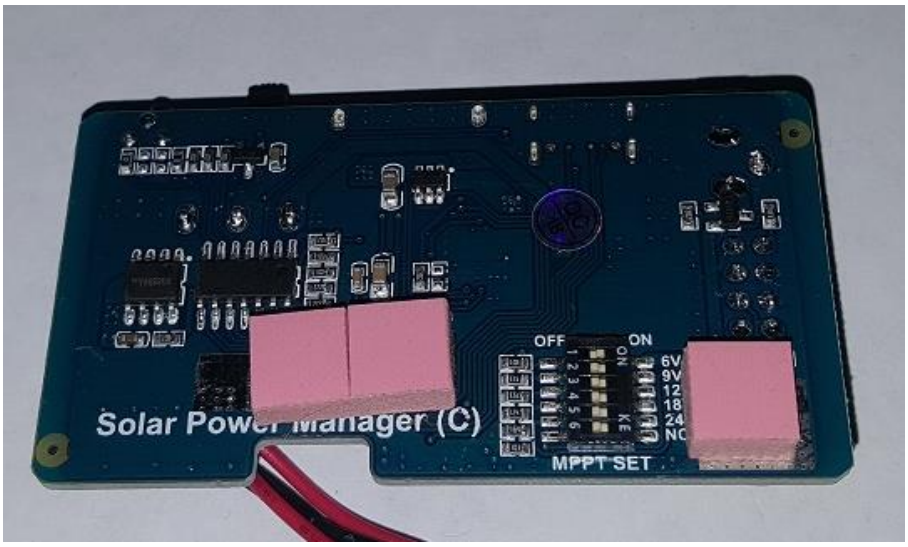


Abbildung 95: Solar Manager Dip-Switch

Ergebnis

Der fertige Prototyp wurde getestet und funktioniert. Die Sensorboxen, die Sensorboardboxen und die Elektronikbox kann eins zu eins so übernommen werden.
Gewisse Anpassungen müssen für den Aufbau in der TEKO dennoch erfolgen.
Die Halterungen für die einzelnen Gehäuse müssen natürlich an den richtigen Standort angepasst werden. Auch muss der Halter für die Solarpanels verstärkt werden, sonst könnte es Probleme geben, wenn es stürmt. Aus diesem Grund wurde der Prototyp erstellt, um solche Problem e zu erkennen und beheben zu können.
Ein Inbetriebnahmeprotokoll sowie eine Auswertung der Arbeit wird erst beim Aufbau bei der TEKO gemacht. Bei der TEKO können nochmals andere Probleme auftauchen wie beim Prototy p.

Der Aufbau bei der TEKO Zürich

Es ist an der Zeit den Aufbau bei der TEKO Zürich zu vollziehen. Dazu müssen die Halter der Sensorboxen, der Sensorboardboxen sowie die ganze Halterung der Elektronikbox und Solarpanels gebaut werden. In diesem Abschnitt folgt die Wegleitung zum fertigen Aufbau bei der TEKO Zürich.

Die Halter bei der TEKO Zürich

Die ganzen Halter werden aus Aluminium hergestellt und mit Edelstahl Schrauben und Muttern befestigt.

Abbildung 96: Sensorboxhalter

Abbildung 97: Sensorboardboxhalter

Der Halter für die Elektronikbox ist der komplexeste, da auch die Solarpanelhalterung auf diesem Halter montiert wird. Hier wurde bedacht, dass die Elektronikbox möglichst gut geschützt ist gegen die Witterungen. Das Gehäuse der Elektronikbox ist wasserdicht, jedoch durch die Anbringung der Ventilatoren mussten Löcher ins Gehäuse gemacht werden. Dadurch sind auf drei Seiten Platten angebracht, um möglichst viel der Witterungen abzuhalten. Ausserdem ist es ein Schutz, wenn mal was an der Elektronikbox gemacht werden muss.
Die beiden Solarpanel müssen bei ihrem Halter eine weitere Stütze erhalten, welche ein biegen durch Stürme verhindert.

Abbildung 98: Elektronikboxhalterung mit Solarpanel Halterung

Die Verdrahtung bei der TEKO Zürich

Die Verdrahtung bei der TEKO konnte ohne weitere Probleme durchgeführt werden. Es wurde darauf geachtet, dass alle Kabel schön und sauber verlegt werden. Alle Kabel haben an den Haltern noch Aluröhren bekommen, welche mit Kabelbinder befestigt wurden. Dies aus dem Grund, falls jemand in die Sensoren fahren würde, kann so ein Halter schnell ersetzt werden. Ausserdem wurden alle Kabel am Boden mit einem blauen Schlauch geschützt, sodass es ins Auge sticht, wenn jemand etwas am Rasen macht und nicht die Kabel zerschneidet.

Die sechs Sensorboxen wurden, wie im untenstehenden Bild aufgebaut und verkabelt.

Abbildung 99: Senorbox bei der TEKO Zürich

Die drei Sensorboardboxen wurden, wie im untenstehenden Bild aufgebaut und verkabelt.

Abbildung 100: Sensorboardbox bei der TEKO Zürich

Die Elektronikbox wie auch das Solarpanel wurden, wie im untenstehenden Bild aufgebaut und verkabelt.

Abbildung 101: Elektronikbox und Solarpanel bei der TEKO Zürich

Nummerierung der Parkplätze bei der TEKO Zürich

Die Parkplätze wurden mit den Sensoren zusammen nummeriert. Das heisst, Parkplatz 1 ist auch Sensor 1 und so weiter bis Parkplatz 6. Ausserdem wurden die Kabel, welche aus der Elektronikbox kommen und an die Sensorboardboxen gehen mit verschiedenen Isolierbandfarben gekennzeichnet. So kann in einem Fehlerfall auch klar erkannt werden, um welche Sensoren es sich handelt.

Abbildung 102: Parkplätze nummeriert.

Kontrolle

Inbetriebnahmeprotokoll

Der Aufbau wird gemäss den Punkten im Pflichtenheft überprüft und getestet.

Inbetriebnahmeprotokoll für smarte Parkplatzüberwachung

					ja	nein
Geräte und Equipment						
	Laptop / Tablett				x	
	Elektronikbox				x	
	Solar Panels				x	
	Sensorboxen				x	
	Sensorboardboxen				x	
	Solar Manager (c)				x	
	Raspberry Pi 4				x	
Überprüfung Funktionen						
Bitte alle Geräte und Equipment ordnungsgemäss verdrahten und starten.						
Bitte überprüfen das alle Ultraschallsensoren keine Hindernisse haben.						
Aufstartzeit Raspberry Pi ca. 1 Minute						
Software Version:	V01.00.02 TEKO (Deutsch)				x	
	V01.00.02 Prototyp (Deutsch)					x
Starten alle Geräte ordnungsgemäss auf und verbinden mit WLAN					x	
IP-Adresse des Raspberry Pi im Browser eingeben http://192.168.73.128:1880/ui					x	
erscheinen die Parkplätze, Temperatur, ESP32 aktiv, Notbelüftung					x	
Reagieren die Parkplätze 1 - 6 auf eine Änderung					x	
Ändert sich die Anzeige im User Interface des jeweiligen Parkplatzes					x	
Verändert sich die Temperatur					x	
Verändert sich die Temperaturfarbe bei folgenden Werten						
	-30 bis 0°C	0 bis 40°C	40 bis 80°C			
User Interface	blau	grün	rot		x	
Schalten die Ventilatoren automatisch bei 50°C ein und bei kleiner 48°C aus					x	
Ventilatoren lassen sich über User Interface steuern und die Anzeige ändert					x	
Schaltet der ESP32 um 21:00 Uhr in den Deep-Sleep und startet 07:00 Uhr wieder					x	
Ändert die Zeitumstellung richtig und sendet Node-RED die aktuelle Zeit					x	
Ändert die Anzeige ESP32 aktiv im User Interface nach 2 Minuten ohne Strom					x	

Ort, Datum
Glattbrugg, 18.10.2022 / Dominik Viola

Ergebnis der Inbetriebnahme

Die Inbetriebnahme konnte erfolgreich durchgeführt werden. Zwei Punkte gab es jedoch, welche erst bei der Inbetriebnahme auftauchten.

Der eine Punkt ist, dass die Sensoren aufgrund der Bodenverhältnisse etwa 30cm weiter hinten als gewünscht platziert werden mussten. Dadurch musste im Node-RED die Distanz für die Parkplätze, wenn diese belegt sind, auf 120cm angepasst werden.

Der zweite Punkt ist, dass das WLAN-Signal während des Tages schwach ist. Um den Feierabend rum, wurde das TEKO WLAN besser. Es wurde zusammen mit Adrian Aegler entschieden einen WLAN-Repeater im Schulzimmer 02 zu verbauen.

Christian Meier stattete nach seiner Arbeit der Inbetriebnahme einen Besuch ab und der fertige Aufbau konnte Ihm demonstriert werden.

Auswertung

Das Projekt konnte umgesetzt werden und mit positivem Abschluss abgeschlossen werden.

Der Zeitplan konnte grösstenteils eingehalten werden. Trotzdem war der Zeitplan mehr als eng gewählt, dies liegt aber auch daran, dass mit Verzug diese Diplomarbeit gestartet werden konnte. Trotz Reserven im Zeitplan konnte dieses Projekt erst in der letzten Woche abgeschlossen werden. Grund dafür war die berufliche Situation.

Die Diplomarbeit konnte mit einem Zeitaufwand von knapp 600 Stunden erarbeitet werden.

Es gibt sicher noch ein paar kleine Verbesserungen, die eingebaut werden könnten.
Das ganze System funktioniert wie gewünscht. Jetzt muss ein Langzeittest zeigen, wo noch weitere Verbesserungen erfolgen müssen.
Ein Beispiel für eine Verbesserung könnte sein, dass nicht minütlich die Sensoren abgefragt werden, sondern zum Beispiel alle 15 Minuten. Dies könnte mit einem weiteren Case im Programmcode des ESP32 erreicht werden. Dazu könnte man den ESP32 in den Light-Sleep versetzten. Dies hätte zur Folge, dass der Stromverbrauch noch ein wenig mehr sinkt.

Leider konnte nicht überprüft werden, wie sich das ganze System bei winterlichen Verhältnissen verhält. Sollte dort ein Problem auftreten, kann jeder Zeit der Solar Manager heraus genommen werden und über die USB-C Schnittstelle extern aufgeladen werden. Zum Beispiel übers Wochenende, wenn kein Schulbetrieb ist.

Persönliches Resümee

Ich habe diese Diplomarbeit als eine sehr interessante Arbeit empfunden, jedoch für mich allein sehr zeitintensiv. Erstens weil die erste Diplomarbeit abgesagt wurde und zweitens wegen den vielen Arbeitsstunden im Betrieb. Die komplette Diplomarbeit wurde in der privaten Freizeit erarbeitet. Ich bin aber trotz allem auf mich selbst stolz, dass ich dieses Projekt bis zum Schluss durchgezogen habe. Es war nicht immer leicht doch als das fertige Produkt bei der TEKO aufgebaut stand war das ein großartiges Gefühl. Diese Diplomarbeit hatte alles drin über Programmieren, Mechanik, Bestückung, Elektronik, Planung und Organisation. Am meisten Freude hatte ich tatsächlich als die Programmierung so funktionierte wie ich es mir vorstellte. Ich fand es eine großartige Arbeit und bedanke mich hier nochmals bei Adrian Aegler fürs Vertrauen und hoffe, dass er auch zufrieden mit der Arbeit ist.

Abbildungsverzeichnis

Abbildung 1: Projektablauf nach IPERKA 9
Abbildung 2: Ultraschallsensor A02YYUW 17
Abbildung 3: Messaufbau mit ESP8266 und Sensor 17
Abbildung 4: Schema ESP8266 und Sensor bei 3.3V 18
Abbildung 5: Parkplatz frei 18
Abbildung 6: Parkplatz besetzt 19
Abbildung 7: Laser Distanz Sensor 20
Abbildung 8: Messaufbau mit Arduino und Sensor 20
Abbildung 9: Schema Arduino Mega 2560 und Sensor 21
Abbildung 10: 6 in 1 USB zu Serial-Wandler 21
Abbildung 11: USB zu RS485-Wandler 22
Abbildung 12: Software mit Befehl 23
Abbildung 13: Aufbau des Lesebefehles 23
Abbildung 14: Aufbau des Rückgabewertes 23
Abbildung 15: Ausgabe der Software 24
Abbildung 16: Laser Distanz Sensor RS485 Seite Top 25
Abbildung 17: Grossaufnahme des Sensors RS485 25
Abbildung 18: Bottom Seite mit RX und TX vorhanden 26
Abbildung 19: Grossaufnahme Sensor UART 27
Abbildung 20: Ausgabe im Serial Monitor 27
Abbildung 21: Ausgabe im Serial Monitor schwarzes Fahrzeug 28
Abbildung 22: Ultraschallsensor JSN-SR04T-V3.0 29
Abbildung 23: Messaufbau mit ESP32 und Sensor 29
Abbildung 24: Schema ESP32 und Sensor bei 3.3V 30
Abbildung 25: Ausgabe auf dem Serial Monitor 30
Abbildung 26: Sensorboard ohne Modifikation 31
Abbildung 27: Sensorboard mit Modifikation 31
Abbildung 28: Arduino Mega 2560 34
Abbildung 29: Arduino Mega 2560 Pinout 34
Abbildung 30: Messaufbau mit Arduino Mega 2560 und Sensor 35
Abbildung 31: ESP8266 mit Base Board Protoshield mit Spannungsregler 36
Abbildung 32: ESP8266 Pinout 36
Abbildung 33: Tabelle mit den nutzbaren GPIO's 37
Abbildung 34: Messaufbau mit ESP8266 und Sensor 38
Abbildung 35: ESP32 S2 Saola 1R 39
Abbildung 36: ESP32 S2 Saola 1 Pinout 39
Abbildung 37: Messaufbau mit ESP32 S2 Saola 1R und Sensor 40
Abbildung 38: Okdo Raspberry Pi 4 Bundle 42
Abbildung 39: Messaufbau mit Raspberry Pi 4 43
Abbildung 40: PCF8574 I2C Erweiterungsboard 44
Abbildung 41: Messaufbau PCF8574 45
Abbildung 42: Temperatursensor AM2302 DHT22 46
Abbildung 43: Messaufbau Temperatursensor 46
Abbildung 44: Relais SRD-05VDC 48
Abbildung 45: Noctua 5V Ventilator 48
Abbildung 46: Messaufbau Relais und Ventilatoren 49

Abbildung 47: Solarpanel....51
Abbildung 48: Solar Power Manager mit Akkus....51
Abbildung 49: Messaufbau Solarpanel und Solar Power Manager....52
Abbildung 50: Testaufbau für die ESP32 Software....56
Abbildung 51: Flussdiagramm Software....57
Abbildung 52: Softwarecode Sensor Parkplatz....59
Abbildung 53: Softwarecode Temperatur....60
Abbildung 54: Softwarecode Zeit....61
Abbildung 55: Softwarecode Deep-Sleep....61
Abbildung 56: Benutzeroberfläche komplett....62
Abbildung 57: Funktion Parkplatz 1 mit debug....64
Abbildung 58: Abfrage Parkplatz frei?....64
Abbildung 59: Parkplätze frei....65
Abbildung 60: Parkplätze besetzt....65
Abbildung 61: Funktion ESP32 aktiv mit debug....66
Abbildung 62: Abfrage ESP32 aktiv?....66
Abbildung 63: ESP32 aktiv....67
Abbildung 64: ESP32 inaktiv....67
Abbildung 65: Temperatur....68
Abbildung 66: Abfrage "nan"....68
Abbildung 67: Temperatur -30 bis 0°C....69
Abbildung 68: Temperatur 0 bis 40°C....69
Abbildung 69: Temperatur 40 bis 80°C....69
Abbildung 70: Notbelüftung und Ventilatoren....70
Abbildung 71: Abfrage Ventilator an?....70
Abbildung 72: Notbelüftung ein, Ventilator aktiv....70
Abbildung 73: Notbelüftung aus, Ventilator inaktiv....70
Abbildung 74: Zeit....71
Abbildung 75: Millisekunden zu Sekunden....71
Abbildung 76: Prototyp....72
Abbildung 77: Elektroschema des Prototyps mit Originalkabelfarben....73
Abbildung 78: Sensorbox vermasst....74
Abbildung 79: Stopfen bearbeitet....75
Abbildung 80: Senorbox....75
Abbildung 81: Senorboardbox vermasst....76
Abbildung 82: Winkel bemasst....77
Abbildung 83: Winkel modifiziert....77
Abbildung 84: Weg zum fertigen Winkel....78
Abbildung 85: Sensorboardbox....79
Abbildung 86: Elektronikbox untere Seite vermasst....80
Abbildung 87: Elektronikbox rechte Seite vermasst....81
Abbildung 88: M4 Gewinde im Boden der Elektronikbox....81
Abbildung 89: Platte Solar Manager vermasst....82
Abbildung 90: Befestigung Solar Manager....82
Abbildung 91: Leiterplatte bemasst....83
Abbildung 92: Leiterplatte komplett bestückt....84
Abbildung 93: Elektronikbox....85
Abbildung 94: Halterung mit Solarpanels....86

Abbildung 95: Solar Manager Dip-Switch 86
Abbildung 96: Sensorboxhalter 88
Abbildung 97: Sensorboardboxhalter 88
Abbildung 98: Elektronikboxhalterung mit Solarpanel Halterung 89
Abbildung 99: Senorbox bei der TEKO Zürich 90
Abbildung 100: Sensorboardbox bei der TEKO Zürich 91
Abbildung 101: Elektronikbox und Solarpanel bei der TEKO Zürich 92
Abbildung 102: Parkplätze nummeriert. 93

Tabellenverzeichnis

Tabelle 1: Funktionale Anforderungen 12
Tabelle 2: Nicht funktionale Anforderungen 13
Tabelle 3: Risikoanalyse 15
Tabelle 4: Sensortypen 32
Tabelle 5: Mikrokontrollertypen 41
Tabelle 6: Raspberry Pi 4 43
Tabelle 7: PCF8574 I/O Erweiterung 45
Tabelle 8: Temperatursensor 47
Tabelle 9: Ventilator und Relais 49
Tabelle 10: Stromaufnahme gesamt 50
Tabelle 11: Solar und Zubehör 54

Linkverzeichnis

- https://www.youtube.com/watch?v=1CM_N2XZH1k
- https://www.youtube.com/watch?v=s_rpogTWl4c
- https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/
- https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/
- https://www.conrad.ch/de/ratgeber/technik-einfach-erklaert/ip-schutzklassen-und-schutzarten.html
- https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/
- https://www.dfrobot.com/
- https://jovanovicmilos.wordpress.com/iperka
- https://wiki.dfrobot.com/Weatherproof_Ultrasonic_Sensor_With_Separate_Probe_SKU_SEN0208
- https://www.google.ch/
- https://www.youtube.com/watch?v=qyqP5zIxryc
- https://randomnerdtutorials.com/esp8266-pinout-reference-gpios
- https://dl.espressif.com/dl/package_esp32_index.json
- https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

Anhängeverzeichnis

Die Anhänge befinden sich auf dem beigelegten USB-Stick in den jeweiligen Ordnern.

- Qualifikationsprofil und Lebenslauf
- Sitzung mit Diplomcoach
- Software
- Stückliste
- Zeitplan
- Datenblätter
- Schema
- Anweisung TEKO